AD-A126 947   PILOT SCHEDULING IN A FIGHTER SQUADRON(U) AIR FORCE   1/2
INST OF TECH WRIGHT-PATTERSON AFB OH W H ROEGE FEB 83
AFIT/CI/NR-83-2T
UNCLASSIFIED   F/G 5/9   NL

Every flight, no matter what type (there are other types of flights with no minimum requirement), will count toward a pilot's total flight requirement. This cost is constant over all flights j for a given pilot i.

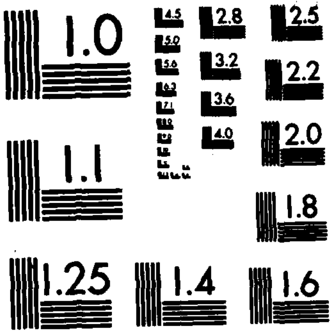
To satisfy the type requirements, flight j must be the same type as the requirement in question. The second cost depends on j, as well as i. Let

c_i^1 = the cost (number of flights flown/total flight requirement) associated with total flights for pilot i,

and

c_{ij}^2 = the costs (number of type j flights flown/type j flight requirements) associated with specific types of flights.

We will associate c_i^1 with arcs s-i since they apply to all flights pilot i flies. Similarly, we associate c_{ij}^2 with arcs i-j since they depend on the type of flight j is. We can weight the components to reflect the scheduler's view of which component is more important relative to the others (i.e. we may want to emphasize the completion of air refueling requirements over air combat training missions). The objective function f(x) can now be written as



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$\text{(BIP)} \qquad \sum_i x_{ii'} + x_{si'} = u_i - l_i \qquad \text{all } i' \qquad (3.23)$$

$$\sum_{i'} x_{si'} = \sum_j b_j - \sum_i l_i \qquad (3.24)$$

$$\sum_j a_{ij}\, x_{ij} = u_i \qquad \text{all } i \qquad (3.25)$$

$$\sum_j f_{ikj}\, x_{ij} \le 1 \qquad k = 1,\ldots, N, \text{ all } i \qquad (3.26)$$

$$x_{ij},\, x_{ii'},\, x_{si'} \ge 0, \text{ integer.} \qquad (3.27)$$

3.4 Example Formulation

Let us illustrate the formulation with a simple example. Consider the hypothetical flight schedule shown in figure 3-5, with six formations, requiring eight pilot assignments. In the example we have four pilots available. Suppose that we require each of the pilots to fly at least one, but no more than three flights. Suppose too, that pilots 2, 3, and 4 are unavailable for flights 2, 6, and 3 respectively. The resulting node adjacency matrix $\{a_{ij}\}$ and time overlap constraint matrix $\{f_{kj}\}$ (constraints (3.26)) are shown in figure 3-6. Note that this matrix is strictly showing the conflicts between flights. We will add the restriction that a pilot i be available and qualified (i.e. $a_{ij} = 1$) at a later time.

AD A126947

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

①

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 83-2T	2. GOVT ACCESSION NO. [illegible]	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Pilot Scheduling in a Fighter Squadron		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William Henry Roege		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Massachusetts Institute of Technology		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE Feb 83
		13. NUMBER OF PAGES 124
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17 12 april 83		Lynn Wolaver LYNN E. WOLAVER Dean for Research and Professional Development AFIT, Wright-Patterson AFB OH
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED		

DTIC
ELECTE
1983
S
E

FILE COPY

83 04 19 006

Day 1	Flight 1	Flight 2	Flight 3
Brief time	0515	0930	1400
Takeoff time	0715	1130	1600
Type flight	Air Combat	DART	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0830	1245	1715
End debrief time	1015	1430	1900

Day 2	Flight 4	Flight 5	Flight 6
Brief time	0500	0930	1400
Takeoff time	0700	1130	1600
Type flight	Air Refuel	Air Combat	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0815	1245	1715
End debrief time	1000	1430	1900

Figure 3-5
Example Problem Schedule

83-2

Pilot Scheduling in a Fighter Squadron

by

William Henry Roege

B.S., United States Air Force Academy
(1976)

DTIC
COPY
INSPECTED
2

Accession For
NTIS GRA&I ☒
DTIC TAB □
Unannounced □
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

SUBMITTED TO THE
SLOAN SCHOOL OF MANAGEMENT
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF THE
DEGREE OF

MASTER OF SCIENCE IN
OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1983

© Massachusetts Institute of Technology 1983

Signature of Author William H. Roege
Sloan School of Management
December 17,1982

Certified by Thomas L. Magnanti
Thomas L. Magnanti
Thesis Supervisor

Accepted by Jeremy F. Shapiro
Jeremy F. Shapiro
Co-Director, Operations Research Center

Flights

	i \ j	1	2	3	4	5	6
	1	1	1	1	1	1	1
Pilots	2	1	0	1	1	1	1
	3	1	1	1	1	1	0
	4	1	1	0	1	1	1

Node-node adjacency matrix

Flight

	k \ j	1	2	3	4	5	6
	1	1	1	1			
	2		1	1			
Flight	3			1	1		
	4				1	1	1
	5					1	1
	6						1

Feasibility constraint matrix

Figure 3-6

Example Problem Data

		Total	ACTT	DART	NINT	AARD
	weight	1	1	1	1	1
	1	1	1	2	0	2
Pilot	2	2	1	N/A	1	0
	3	2	0	1	0	0
	4	1	0	0	0	1

Completion percentages
(in decimal form [i.e. 1 = 100%])

note: a large weight deemphasizes the type of flight, here we weight all types evenly

Flight

	i \ j	1 ACTT	2 DART	3 NINT	4 AARD	5 ACTT	6 NINT
	1	2	3	1	?	2	1
Pilot	2	3	☒	3	2	3	3
	3	2	3	2	2	2	☒
	4	1	1	☒	2	1	1

Example cost matrix

Figure 3-7
Example Problem Costs

PILOT SCHEDULING IN A FIGHTER SQUADRON

by

WILLIAM HENRY ROEGE

Submitted to the Sloan School of Management
on January 13, 1983 in partial fulfillment of
the requirements for the Degree of Master of Science
in Operations Research

ABSTRACT

Air Force fighter pilots, in order to remain combat qualified, must complete flight training every 6 months as specified by Tactical Air Command Manual (TACM) 51-50. Presently, scheduling is manual. As a result, pilots do not receive an optimum flow of training and often do not complete their required training.

We propose a computer model, an integer program, based on branch and bound techniques to solve the problem on a micro-computer. The model includes complicating constraints such as crew rest restrictions and absences from duty and ensures that each pilot receives at least a minimum, or no more than a maximum, number of flights per week.

Our method involves relaxing some of the constraints (e.g. crew rest constraints) to obtain a network flow problem. We tighten the relaxation by solving small set covering problems derived from the relaxed constraints.

The model was developed and tested on an IBM personal computer.

Thesis Supervisor: Prof. Thomas L. Magnanti

Title: Professor of Operations Research

Pilot 1							Pilot 2						Pilot 3						Pilot 4						s				
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{11}'	x_{21}	x_{23}	x_{24}	x_{25}	x_{26}	x_{22}'	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{33}'	x_{41}	x_{42}	x_{44}	x_{45}	x_{46}	x_{44}'	x_{s1}'	x_{s2}'	x_{s3}'	x_{s4}'	RHS
1	3	0	2	1	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	2	2	1	Z
1							1						1						1										2
	1													1						1									1
		1						1							1														1
			1						1							1					1								2
				1						1							1					1							1
					1						1												1						1
						1																			1				2
												1														1			2
																		1									1		2
																								1				1	2
																									1	1	1	1	4
1	1	1	1	1	1	1																							3
							1	1	1	1	1	1																	3
													1	1	1	1	1	1											3
																			1	1	1	1	1	1					3
1	1	1																											1
	1	1																											1
		1	1																										1
			1	1	1																								1
				1	1																								1
							1	1																					1
								1	1																				1
									1	1	1																		1
										1	1																		1
													1	1	1														1
														1	1														1
															1	1													1
																1	1												1
																			1	1									1
																					1	1	1						1
																						1	1						1

Figure 3-8

BIP Formulation of the Example Problem

ACKNOWLEDGEMENT

I would like to thank the United States Air Force and the Tactical Air Command for selecting me for the Senior Commander's Program, and for giving me the opportunity to study at MIT.

I would also like to thank the faculty and staff at the Operations Research Center for their help in making my stay at MIT both fruitful and rewarding.

I particularly wish to thank Professor Tom Magnanti for his guidance during the course of this thesis, and for unselfishly giving his time and energy near the end of the semester when everything was hectic.

Finally, I wish to thank my wife, Lynda, for sticking with me during the past year and a half.

WHR

To help understand f_{kj} consider row 1 in $\{f_{kj}\}$. The first three 1's mean that flight 1 conflicts with flights 2 and 3. Row 3 shows that flights 3 and 4 conflict (because of overnight crew rest), and row 4 shows that flights 4, 5, and 6 conflict.

To develop the cost matrix $\{c_{ij}\}$ we assign weights, as shown in figure 3-7, to the cost components, and multiply them by the hypothetical completion percentages (also in figure 3-7). The resulting cost matrix is the last matrix depicted in figure 3-7.

As an example, consider pilot 1 and flight 2. The cost c_{12}, is the weight for total flights (1), times the completion percentage of total flights for pilot 1 (100 per cent = 1), plus the weight for DART missions (1), times the completion percentage (2), which is $1 \cdot 1 + 1 \cdot 2 = 3$. So $c_{12} = 3$, as shown in figure 3-7.

Figure 3-8 shows our sample problem, expanded in the form of (BIP). The first 6 constraints are the node balance constraints for the flights. The second 4 constraints are for the i' nodes. The next 5 constraints are for s and the pilots. The last 16 constraints are from the $\{f_{jk}\}$ matrix, but are now adjusted for the individual pilots. We will use a portion of this problem to illustrate the solution

Table of Contents

Title Page..1

Abstract..2

Acknowledgement...3

Table of Contents...4

List of Figures...6

Chapter 1: Introduction.......................................7

Chapter 2: The Fighter Squadron..............................10

2.1 Training...10

2.1.1 Types of Training......................................10

2.1.2 Training Requirements..................................11

2.1.3 Pilot Qualifications...................................12

2.1.4 Continuity and Crew Rest...............................14

2.2 Scheduling...15

2.3 The Model..18

2.3.1 The Goals of the Model.................................18

2.3.2 Costs of the Problem...................................19

2.3.3 Problem Constraints....................................20

Chapter 3: Problem Formulation...............................22

3.1 Network Constraints......................................23

3.1.1 Developing the Constraints.............................23

3.1.2 Eliminating the Supply Bounds..........................27

3.2 Complicating Constraints.................................33

3.2.1 Overlap Constraints....................................33

3.2.2 Crew Rest Constraints..................................34

procedure in chapter 5.

CHAPTER 4

REVIEW OF THE LITERATURE

Scheduling has many applications. One major application, job shop and machine job scheduling problems

Arabeyre, Fearnley, Steiger, and Teather (1) survey the early attempts to solve the airline pilot scheduling problem. Most researchers separated the problem into two parts, (1) assigning flight legs (one takeoff to one landing) to rotations (a round trip of one to three days), and (2) assigning pilots to the rotations. The first problem attempted to minimize "dollar" costs, such as costs of overnight lodging. The second problem aimed to distribute pilot monthly flight time evenly.

Usually researchers and practitioners considered the first problem to be the most difficult since it had to deal with complicating constraints due to union rules, FAA regulations, and company policies. Etcheberry (11) developed an implicit enumeration algorithm, using a branch and bound framework with Lagrangian relaxation, to solve large set covering problems such as this one.

Rubin (36) solved the problem by reducing the number of constraints as much as possible before solving it. He would then consider subsets of the constraint matrix columns, find the best solution over that subset, and repeat the process until obtaining a satisfactory solution.

Marsten (26) developed an algorithm to solve the related set partitioning problem. This algorithm ordered the

List of Figures

Figure 2-1: Example Flight Formations....13

Figure 2-2: Typical Day's Schedule....16

Figure 3-1: Assignment Network....24

Figure 3-2: Eliminating Lower Bounds....29

Figure 3-3: Eliminating Upper Bounds....30

Figure 3-4: Network After Bound Elimination....31

Figure 3-5: Example Problem Schedule....42

Figure 3-6: Example Problem Data....43

Figure 3-7: Example Problem Costs....44

Figure 3-8: BIP Formulation of the Example Problem....45

Figure 5-1a: Problem Structure....56

Figure 5-1b: Time Overlap Constraint Structure....57

Figure 5-1c: Feasibility Constraint Matrix....58

Figure 5-2: Branch and Bound Flow Chart....70

Figure 5-3a: Network Representation of the Example Problem....72

Figure 5-3b: Example Problem Schedule....73

Figure 5-3c: Example Problem Cost and Time Constraint Matrices....74

Figure 5-4a: Example Problem--First Solution....76

Figure 5-4b: Solution After First Reassignment....77

Figure 5-5a: Example Problem--Second Solution....79

Figure 5-5b: Solution After Second Reassignment....80

Figure 5-6: Example Problem--Third Solution....81

Figure 5-7: Branch and Bound Summary....82

constraint matrix lexicographically before starting the optimization. The algorithm then takes advantage of the constraint structure to help fathom candidate problems quickly.

Garfinkel and Nemhauser (15) developed a set-partitioning procedure that reduces the problem size by eliminating row and column vectors before applying their algorithm. The algorithm then orders the data so the rows with the least number of non-zero entries appear first, and the columns with the lowest costs are on the left. They then use an implicit enumeration algorithm that takes advantage of this structure to build possible solutions. Pierce (33) independently developed a similar algorithm.

Nicoletti (32) viewed the second problem (assignment of pilots to rotations) as a network assignment problem and successfully used the out-of-kilter method to find solutions.

The fighter pilot scheduling problem differs from the airline scheduling problem in that all the fighter flights originate and terminate at the same base. This eliminates the need to develop rotations, although we still must deal with crew rest and other regulations, just as the airlines must.

A possible formulation of the fighter pilot scheduling problem is in the form of "k-duty period" scheduling problem. This problem deals with schedules consisting of k independent contiguous scheduling periods; for example, a schedule might assign a person to work k four hour shifts each separated by two hour breaks.

Shepardson (40) deals with this problem. The general idea is to start with a proposed (yet feasible) subset of schedules as the columns of a constraint matrix, with its rows being the jobs to be filled. He then separates the columns into new columns each with only one contiguous scheduling period. For example, he would separate a 2-duty schedule containing two 4-hour shifts into two columns each representing a single 4-hour shift. This solution strategy is attractive because problems in which all schedules have a single contiguous duty can be solved as network flow problems.

He then adds extra side equations to ensure that if a new column is in the solution, then all the new columns associated with its column in the original problem formulation, are also in the solution. In our example, if one of the two columns with the 4-hour shifts is in the solution, then they both must be in the solution. He then dualizes these side equations and uses Lagrangian relaxation

CHAPTER 1

INTRODUCTION

The recent development of the Rapid Deployment Force and the events in the Falkland Islands (22) underscore the necessity for our combat forces to be ready at a moment's notice. To do its part, the Tactical Air Command (TAC) must be ready to fly anywhere at a day's notice. To achieve this capability, TAC must maintain a high level of training for all of its pilots. The Air Force has many levels of command starting with the President, Department of Defense, and Headquarters Air Force. Although it has four major commands with tactical fighters, we will restrict our attention to TAC which commands the fighter units in the continental United States. Under TAC are a number of flying Wings. Each Wing consists of three squadrons of 18 to 30 aircraft. A Wing will normally be assigned to one base, and usually is the only Wing at the base. The squadron is the smallest administrative unit.

The squadron's job is to be combat ready at all times, but strategic decisions on resource allocation are all made well above the squadrcn and Wing levels. For example, the higher authorities determine the number of aircraft in each squadron, the number of pilots assigned to the squadron, and methods to solve the problem.

4.2 Lagrangian Relaxation

In problems with many complicating constraints, Lagrangian relaxation techniques that exploit underlying problem structure (like the single-duty problem that can be solved as a network flow problem) have so far yielded very good results for a wide variety of applications. Fisher (12), Magnanti (25), and Shapiro (39) all give good surveys of Lagrangian relaxation methods, and mention a number of application areas.

Lagrangian relaxation methods attempt to simplify the problem by dualizing some constraints, multiplying them by Lagrange multipliers, and adding them to the objective function. Given a set of multipliers, the relatively easy relaxed problem is solved. Then given the new solution to the relaxed problem, we solve for new multipliers. (In section 5.2 we describe the multiplier selection procedure in more detail.) We can embed this method into a branch and bound framework to systematically exhaust all possibilities, and find the optimal solution. (See (7) and (12) for an explanation of branch and bound methodology.)

the amount of gasoline allocated to the squadron.

TAC also has training guidelines that set the semi-annual training requirements for all pilots. These guidelines are documented in TAC Manual 51-50 (41). TACM 51-50 is written to ensure that all pilots in every squadron are obtaining at least a minimum amount of proper training.

The task for the squadron, then, is to allocate its given resources to ensure that each pilot receives his required training. This may not seem difficult to accomplish, but at the present time with manual scheduling, and with a wide range in training needs for the pilots, many pilots either do not complete their semi-annual requirements, or barely finish in the last week. This invariably leads to "crisis management".

Our proposal is to build a computer model to do much of the routine scheduling, so that schedulers can devote more time to specialized problems. The program will use TACM 51-50 requirements to form an objective function. It will define costs in terms of a percentage of a requirement completed, and will try to schedule pilots who are behind schedule (relative to others) more often than the pilots who are ahead. We will focus on an F-15 air-to-air squadron as a specific application.

Several methods have been proposed to solve for the Lagrange multipliers. The most popular method is subgradient optimization. The method starts with a proposed solution for the multipliers, then uses a subgradient of that solution to move to a better solution. Held, Wolfe, and Crowder (18) give a comprehensive explanation and evaluation of subgradient optimization. Other methods include generalized linear programming (25), the BOXSTEP method (19), dual ascent (10), and so called multiplier adjustment methods (10,13). None of these methods has performed as well as subgradient optimization so far on a wide variety of problems, though multiplier adjustment methods have proved to be successful on facility location problems (Erlenkotter [10]) and generalized assignment problems (13).

Ross and Soland (35) proposed a heuristic for finding multipliers when solving the generalized assignment problem. They relax the supply node bounds and solve the relaxed assignment problem. Their method then assigns multipliers based on the minimum penalty (increase in cost) incurred to make the relaxed solution feasible. For each supply node whose supply bound is exceeded, they find a new assignment that makes that node supply feasible with minimal cost increase. The increase in cost for that node is its new multiplier. These multiplier problems are in the form of knapsack problems (i.e. integer programs with only one

Chapter 2 gives more detail of the training requirements, and the scheduling process. It also defines the goals, costs, and constraints that affect this problem.

Chapter 3 develops the mathematical model for the flying portion of the schedule as an assignment problem.

Chapter 4 reviews the literature related to our scheduling problem.

Chapter 5 discusses solution techniques for the problem, and illustrates our procedure with a small example.

Chapter 6 describes the computer implementation issues, and the computational results.

We have developed a branch and bound algorithm, based on an algorithm proposed by Ross and Soland (35), which solves the scheduling problem we propose. We were successful in coding the algorithm onto an IBM personal computer.

constraint). Until multiplier adjustment methods were developed, their method seemed to be faster than any other for solving generalized assignment problems, their advantage being the ability to quickly solve the small knapsack problems to find the multipliers. We give a more detailed explanation of this procedure in section 5.3.

Fisher, Jaikumar, and Van Wassenhove (13) have developed a new multiplier adjustment method for the generalized assignment problem, which seems to outperform the Ross and Soland algorithm. They start with the Ross and Soland multipliers, and adjust them one by one to eventially obtain a feasible solution to the original problem. Each adjustment ensures that the original problem is closer to feasibility, and eventually the method will yield a feasible solution. Even though it takes much longer to find the multipliers, the method decreases the number of problems it must solve in the branch and bound framework, and therefore runs in less time. We will discuss this method further in section 5.3.

Chapter 5 will apply the techniques discussed here to the fighter pilot scheduling problem.

CHAPTER 2

THE FIGHTER SQUADRON

This chapter focuses on the background necessary to understand the problem, including the training required in TACM 51-50 and the present scheduling system. The second section defines the goals, objectives, costs, benefits, and constraints that relate to the problem, and that underscore the mathematical model that we shall study.

2.1 ɪraining

2.1.1 Types of Training

The squadron administers two types of training. The first is upgrade training and the second is continuation training. Upgrade training is conducted according to a very strict and controlled syllabus, and applies to pilots becoming initially combat qualified (called Mission Ready, or MR). It also applies to those who are training to become flight leads and instructors. Continuation training, on the other hand, entails more flexible requ'rements that must be accomplished every six months (January to June, and July to

CHAPTER 5

SOLUTION PROCEDURES

This chapter will discuss solution methods applicable to the fighter pilot scheduling problem. We will discuss the general problem structure, Lagrangian relaxation solution techniques, the technique developed by Ross and Soland, and methods for solving the unconstrained assignment problem and set covering problems.

5.1 Problem Structure

As we noted in chapter 3, (BIP) is basically a transportation problem with complicating constraints representing time overlap and crew rest restrictions. The problem has the classical primal block angular structure (7) shown in figure 5-1a. The common constraints represent the transportation problem, and the overlap constraints form separable subproblems.

The time constraints also have a special structure. Figure 5-1b shows an enlargement of the shaded block in figure 5-1a. All non zero entries lie between the diagonal

December, called halves). All mission ready pilots participate in this training.

Normally the squadron closely monitors upgrade training and assigns students and instructors to specific flights that meet their needs for a particular mission. Therefore, instructor and student scheduling for upgrade training is essentially fixed, and we concentrate on scheduling only continuation training.

2.1.2 Training Requirements

As mentioned before, TACM 51-50 is the training bible for the squadron. There are three general types of requirements: number of (1) total flights, (2) special types of flights, and (3) specific events to accomplish while flying. We need only concern ourselves with the first two categories since the pilots should be able to perform all their required events as long as we schedule them for their required flights. Appendix A describes each type of flight and its semi-annual requirement.

In addition to flying, the pilots must complete 12 hours of simulator training per half. The squadron must also man other flying related duties. These include Supervisor of

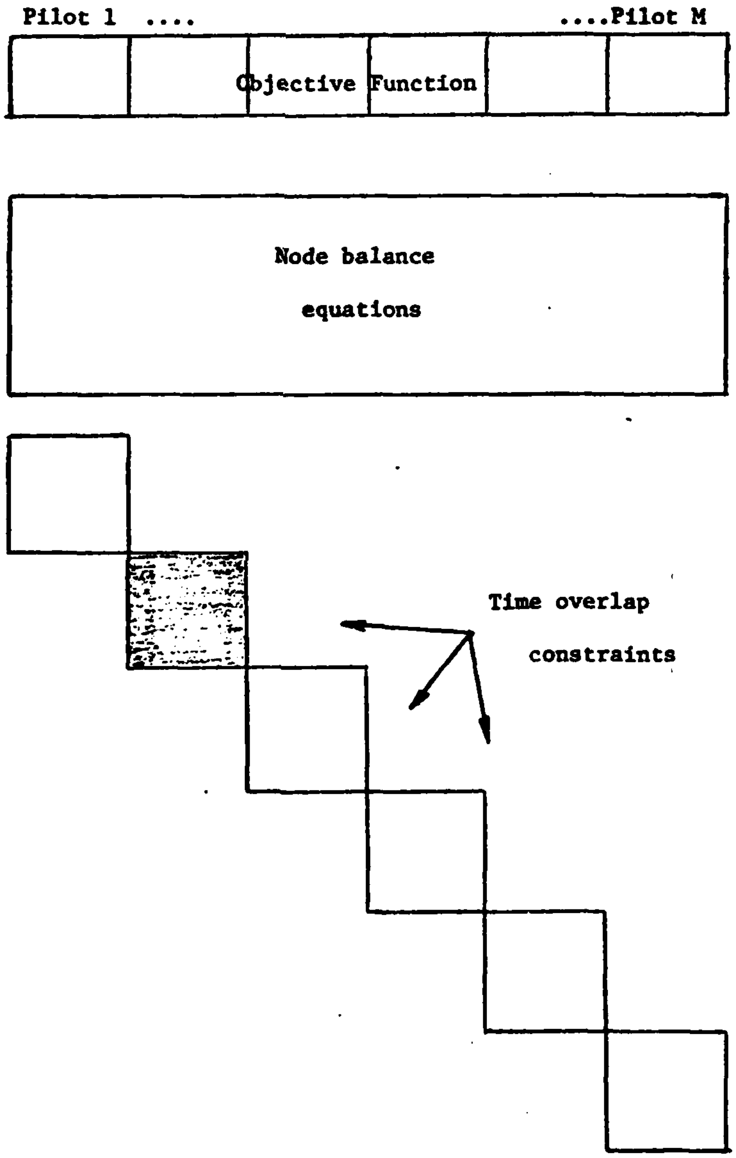


Figure 5-1a
Problem Structure

Flying (SOF), Runway Supervisory Officer (RSO), and Range Training Officer (RTO). Appendix B briefly explains these duties.

2.1.3 Pilot Qualifications

Flying training, as well as combat, is conducted in flights of 2 to 4 aircraft. Each position in the flight requires a minimum qualification. All pilots fit into one of these four qualification categories and are assigned slots in the flight accordingly. These categories are:

1. Instructor pilots (IP)- the most experienced pilots, whose job it is to teach all upgrade training. They also can fly any other position available.
2. Flight leads (FL)- are qualified to lead any flight. They are responsible for continuation training in their flight. They may also fly as wingmen.
3. Wingmen (WG)- are fully combat qualified, but must fly with a flight lead when there is more than one aircraft in a flight.
4. Mission Qualification Trainees (MQT)- are not combat qualified, and may only fly with instructors.

Figure 2-1 shows some normal formations in the air of 2 and 4

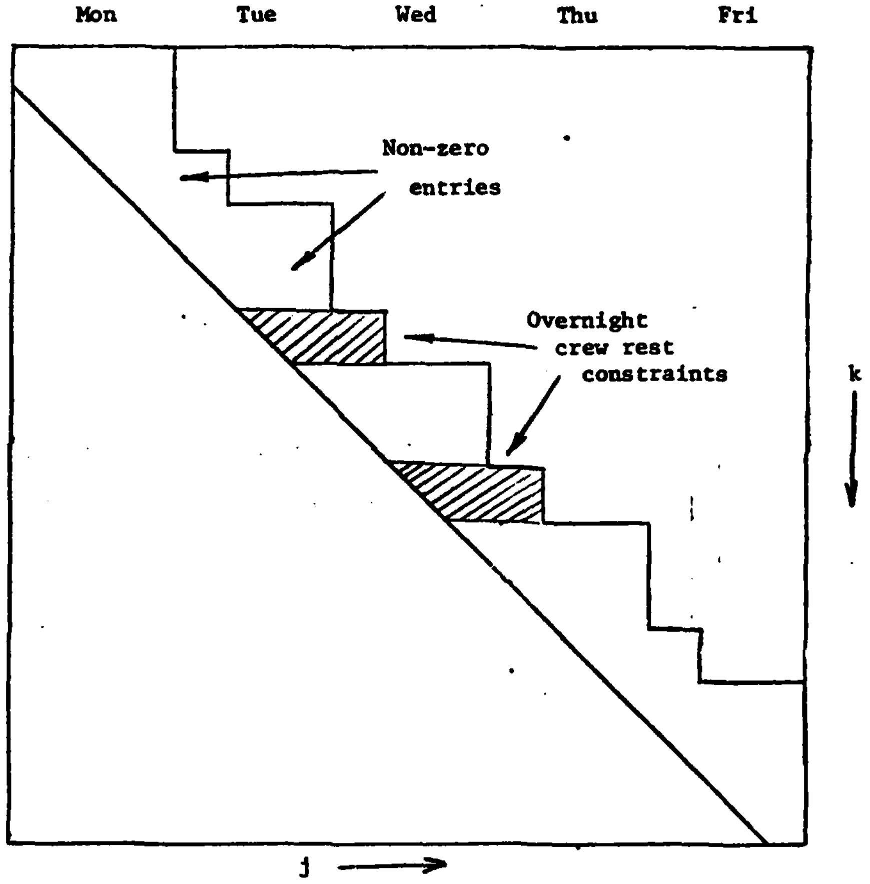


Figure 5-1b
Time Overlap Constraint Structure

Flights

	Day 1			Day 2		
k \ j	1	2	3	4	5	6
1	1	1	1			
2		1	1			

"Bump" from
overnight crew

line and the staircase within the matrix. The shaded "bumps" represent the crew rest constraints that link one day's schedule to the next. If the overnight crew rest constraints weren't present, the subproblems would separate further into daily subproblems. For example, figure 5-1c shows the time constraints for one pilot in the example problem developed in section 5.3. The arrow shows the "bump" resulting from the overnight crew rest constraint. If flights 3 and 4 didn't conflict, then the constraints for day 1 and day 2 would be separable.

5.2 Lagrangian Relaxation

We could conceivably attempt to use general purpose integer programming algorithms to solve this problem, but because of the complexity of the time constraints, these methods probably would not be very efficient. This brute force approach does not take advantage of the network structure in the common constraints, which we can exploit to solve the problem much more efficiently. By using a Lagrangian relaxation algorithm, we can take advantage of the network structure and decrease our solution times.

Fisher (12), Magnanti (25), and Shapiro (39) give a good description of the Lagrangian technique and give many ship flights (the triangles represent aircraft). It also shows we must pair a flight lead or instructor with every wingman. Continuation training involves only flight leads and wingmen, so we only concern ourselves with these two catagories in our study.

2.1.4 Continuity and Crew Rest

Before moving on to the scheduling system, we briefly explain the concepts of continuity and crew rest. Continuity is important because a pilot will become rusty, or at least not fly at his best, with as little as one week without flying. Therefore the squadron will want all available pilots to fly some minimum number of flights each week, depending upon how many flights are available.

"Crew rest" is designed to avoid pilot fatigue. Crew rest has two components. The first component keeps the duty day from being too long. The duty day is measured from the start of the pilot's first duty (flight brief time, or start of a SOF or RSO tour of duty) to the end of his last flying duty (flight landing time, or end of a SOF or RSO tour of duty). The duty day can be no longer than 12 hours.

The second component of the crew rest is designed to citations to applications of this methodology. We will give a general overview here as it relates to the fighter pilot problem.

Lagrangian relaxation is used to provide bounds in a branch and bound algorithm by dualizing some of the constraints. Typically, this procedure is used by constructing a Lagrangian problem that is much easier to solve than the original problem.

In our case we can dualize the node balance constraints, associating Lagrange multipliers v_j with the sink node equations, and multipliers w_i with the supply node equations, giving the "Lagrangian relaxation" problem

$$Z(v,w) = \min \sum_i \sum_j (c_{ij}x_{ij}) + \sum_j v_j (b_j - \sum_i a_{ij}x_{ij}) + \sum_i w_i \quad (u_i - \sum_j a_{ij}x_{ij}) \qquad (5.1)$$

subject to

$$\sum_j f_{ikj}x_{ij} \le 1 \qquad k = 1,\ldots, N, \text{ all } i \quad (5.2)$$

$$x_{ij} \text{ integer.} \qquad (5.3)$$

We can rewrite the objective function as ensure the pilots obtain enough sleep and time to relax. It is the time between the end of the last duty (end of the flight debrief or end of a SOF or RSO tour) one day until the start of the first duty the next. This component must be at least 12 hours.

2.2 Scheduling

The squadron schedulers are a group of three to five pilots. They are responsible for developing the schedule, for deciding the timing and types of flights, and for assigning pilots to those flights. Before they can assign pilots there must be a mission schedule, such as the one in figure 2-2. Each blank, in figure 2-2 represents a slot that needs to be filled by a pilot who is qualified to fill that slot. The flight lead briefs the flight two hours prior to takeoff, and debriefs the flight after it lands (approximate times are indicated).

The mission schedule is heavily influenced by factors exogeneous to the squadron including maintenance's ability to provide aircraft, FAA airspace availability, and availability of other aircraft such as air refueling tankers. The schedulers juggle these factors to design a schedule that shows the mission times, airspace, and mission type.

There are a few methods available for solving for v or w in maximizing Z(v,w). These include subgradient optimization (18), generalized linear programming (for the LP dual problem of maximizing Z(v,w)) (25), and the multiplier adjustment method (10,13). Subgradient optimization has been the dominant procedure used so far, but the new multiplier adjustment method used by Erlenkotter (10) and by Fisher, et al. (13) seems to work much faster in some applications.

The multiplier adjustment method starts with any values of the Lagrange multipliers v and w, which might give a fairly loose lower bound on Z. Then by adjusting each multiplier one by one, we obtain a feasible solution with a much sharper lower bound. This sharper lower bound tends to fathom candidate problems faster than the Ross and Soland method, which we discuss next. See the references for explanations of the procedures discussed so far.

In the next section we discuss a branch and bound method, related to Lagrangian relaxation, developed by Ross and Soland.

5.3 Branch and Bound Algorithm

To solve (BIP), we will use a relaxation algorithm

Times			
Brief	0430	0800	1330
Takeoff	0630	1000	1530
Type flight	ACTT	MQT/ACTT	ACTT
	FL ______ WG ______ FL ______ WG ______	IP ______ MQT ______ FL ______ WG ______	FL ______ WG ______ FL ______ WG ______
Debrief end	0930	1300	1830 (Land 1650)
Brief	0500	0830	1430
Takeoff	0700	1030	1630
Type flight	MQT/INT	DACT	NINT
	IP ______ MQT ______ FL ______	FL ______ WG ______	FL ______ WG ______ FL ______ WG ______
Debrief end	1000	1330	1930 (Land 1750)
Brief	0510	0900	
Takeoff	0710	1100	
Type flight	ACTT	DACT	
	FL ______ WG ______ WG ______	FL ______ WG ______	
Debrief end	1010	1400	

Figure 2-2

Typical Day's Schedule

adapted from Ross and Soland (35). Their algorithm is designed to solve the generalized assignment problem. Our problem structure is such that we can use a slightly modified version of the the algorithm.

5.3.1 Branch and Bound--General

Before discussing the specific aspects of the Ross and Soland method, we review the general principles of branch and bound methods. The general idea is to implicitly enumerate all possible solutions to a problem (such as (BIP)) by cutting the problem in half at each branching step, and then finding the optimal feasible solution for each half.

For instance, we solve a relaxed problem, such as (NET), and find the resulting x^* to be infeasible to (BIP). We select a variable, x_{branch}, to branch on, and split all possible solutions into 2 sets. One set will include all possibilities where $x_{branch} = 1$, and the other set will include all possibilities where $x_{branch} = 0$.

We then solve (NET) again with the stipulation that $x_{branch} = 1$. If the resulting solution is feasible to (BIP) then we know we have the best solution for the $x_{branch} = 1$ branch, and we can focus attention on the solutions where

$x_{branch} = 0$.

We then go to (NET) again and solve it when we set $x_{branch} = 0$. Suppose the new solution is not feasible to (BIP). Then we can repeat the branching process on another separation variable. We still include the restriction of $x_{branch} = 0$ along with any new restrictions.

If during this process, any solution to the relaxed problem has an objective value greater than the value of the best feasible solution found so far, we can stop looking for the optimal solution on that the search on a branch. This process of ending branch is called fathoming.

To find the optimum solution to (BIP), we use the branch and bound method until we have fathomed all possible branches. The lowest cost, feasible solution will then be the optimal solution to (BIP).

5.3.2 Ross and Soland Method

This algorithm utilizes a branch and bound framework that first relaxes the time overlap constraints and then solves the network constraints to obtain a candidate solution x^*. It then forms small integer problems from the violated time constraints, and solves them to find lower bounds and

The pilot schedule is done one week at a time. Scheduling for a longer period would be fruitless, as the schedule is almost never completed exactly as planned. Various factors precipitate change. These include weather cancellations, maintenance problems, pilot illnesses, and unexpected pilot unavailabilities. The daily schedule often differs greatly from the weekly schedule because of these changes. The weekly schedule serves as a basis for the daily schedules and lets pilots know what to expect for the week. If there are no aircraft cancellations or other problems, then the daily and weekly schedules should match.

One of the problems with the manual scheduling system is that with 30 to 40 pilots, each of whom have different requirements, it is very difficult to keep track of everyone. TAC has used a system called TAFTRAMS to monitor the pilots' status, and give schedulers the information they need for assigning pilots to flights.

TAFTRAMS required punch cards to be sent to another building to be entered into a computer. Twice a week a computer generated printout was sent to the schedulers. Thus information was normally 1 to 3 days late. TAFTRAMS would make a squadron-based computer scheduling program difficult to implement.

separation variables to use in the branching process. We use the separation variables to form candidate problems in which we divide the possibilities in half by adding the constraint that the separation variable must be 1 in our next solution. If the next solution to (NET) (or (BIP)) is feasible, then we try the other half of the possibilities (i.e. solve (NET) when the separation variable is fixed at 0). We first discuss the procedure, then illustrate it with the small example problem formulated in chapter 3.

The relaxed problem is

$$z_R = \min \sum_i \sum_j c_{ij} x_{ij} \tag{5.6}$$

subject to

$$\sum_i a_{ij} x_{ij} = b_j \qquad \text{all } j \tag{5.7}$$

$$\sum_{j'} x_{sj'} = \sum_j b_j - \sum_i l_i \tag{5.8}$$

$$\text{(NET)} \qquad \sum_i x_{ij} + x_{sj'} = u_i - l_i \tag{5.9}$$

$$\sum_j a_{ij} x_{ij} = u_i \qquad \text{all } i \tag{5.10}$$

$$x_{ij},\ x_{ij'},\ x_{sj'} \text{ integer} \tag{5.11}$$

which is a min-cost flow transportation problem. Later in

The new system is called AFORMS. It will use a micro computer in the squadron to store TACM 51-50 information, and allow the schedulers access to current information. AFORMS allows us to build a program that uses current information in the squadron computer.

The current manual scheduling system has other problems besides the lack of timely information. There is no central place to keep information concerning when pilots have meetings, appointments, or are on vacation. Sometimes this results in someone being scheduled to fly when he is not available. Crew rest violations occur mainly when the schedule is changed at the last minute, without checking the new pilot's crew rest status.

2.3 The Model

Now that we have an idea of the scheduling situation in the squadron, let us look at how we might go about building a model. First, before considering the mathematical development in chapter 3, let us describe the goals of the model, the relevant cost structures, and the constraints.

2.3.1 Goals of the Model

the chapter we describe methods for solving (NET).

Let x^* denote an optimum flow vector for (NET) and let Z_R denote its optimum objective value. If x^* is feasible for the time constraints, then it is optimal for the original pilot scheduling problem (12).

If the solution x^* to (NET) is infeasible to (BIP), we can then form auxilary problems (subproblems) with the time constraints. We will have one subproblem for each pilot i. The objective of these subproblems is to find the minimum cost reallocation of flights from pilot i to other pilots, so that pilot i's schedule is feasible. By solving these subproblems for all i, we will find a lower bound for Z in (BIP). This lower bound will help fathom the current candidate problem, and help find a separation vari‑ble (to use for the next branch).

Let $\bar{c}_{qj}$ be the reduced cost of the pairing of pilot q to flight j in x^*. Let $\bar{c}_{rj}$ be the next larger reduced cost for flight j, and define

$$p_j = \{\bar{c}_{rj} - \bar{c}_{qj}\},$$

then p_j represents the minimum penalty for reassigning flight j with respect to the solution x^*. Also let

$$J_i = \{j : x^*_{ij} = 1\},$$

and

In general, we want to maintain the virtues of the present system, while using the computer to help alleviate some of the problems now encountered. Therefore to accomplish this goal, the model must:

1. Ensure that TACM 51-50 requirements are met and are being allocated evenly.
2. Ensure that every available pilot flies the minimum number of flights every week.
3. Find a solution to the weekly (and daily) schedules with no crew rest violations or unavailable pilots assigned to duties.
4. Solve the problem in less time than the present system.
5. Be able to run the program on a micro computer.

In addition, the model should provide the means to schedule the flying related duties, and be able to display who is available in case last minute problems arise.

2.3.2 Costs of the Problem

The costs in this problem cannot be measured directly in dollars and cents, although in the long run better training will result in a more cost effective force. The costs here are training costs associated with TACM 51-50 requirements.

$$y_{ij} = \begin{cases} 1 & \text{if we reassign flight } j \text{ from pilot } i \\ & \text{to pilot } r \\ 0 & \text{otherwise.} \end{cases}$$

Consider the problem

$$z_i = \min \textstyle\sum_{j \in J_i} p_j \; y_{ij} \qquad (5.12)$$

subject to

$$(\text{SIP}_i) \qquad \textstyle\sum_{j \in J_i} f_{ikj} y_{ij} \geq d_{ik} \qquad \text{all } k \qquad (5.13)$$

$$y_{ij} = 0 \text{ or } 1, \qquad (5.14)$$

where

$$d_{ik} = \textstyle\sum_j f_{ikj} \; x^*_{ij} - 1.$$

The value of d_{ik} is the minimum number of flights which must be reassigned to satisfy constraint k. The solution, y^*, this problem represents decisions to as to whether to let pilot i keep flight j (i.e. $y^*_{ij} = 0$), or to reassing flight j to pilot r (i.e. $y^*_{ij} = 1$).

If $y^*_{ij} = 0$, then p_j is large, and we would want to keep this pairing as it is. On the other hand, if $y^*_{ij} = 1$ and p_j is small, we will not be hurt much by reassigning flight j to pilot r.

When we solve (SIP_i) the resulting z_i represents the minimum increase in cost by changing x^* to make pilot i's schedule feasible. The overall minimum penalty is $\sum_i z_i^*$, so a lower bound, LB, on (BIP) is

$$LB = Z_R + \sum_i z_i.$$

We can use LB to fathom nodes in the branch and bound procedure (35).

As in Ross and Soland, we can use the solutions y_{ij}^* to suggest a new solution that tends to be feasible. To form the new test solution, we start with the solution x^* from (NET). We then change the x corresponding to $y_{ij}^* = 1$ to zero, and set the corresponding variables variables x_{rj} to one. If this new solution is feasible its objective value is given by LB. The solution is also optimal for the candidate problem we are investigating, since we found the minimum increase in cost when solving the subproblems.

If the new solution is still infeasible, we need to find a separation variable (x_{ij}). A logical choice is one of the variables with $y_{ij}^* = 0$. We choose to branch on the x_{ij} with the maximum p_j for all i. When we branch we will set $x_{ij} = 1$ as the first candidate problem, and $x_{ij} = 0$ as the second.

5.3.3 Algorithm Summary

rules.

Here the squadron restrictions will set only minimum and maximum number of flights per week. There are, of course, many possibilities for other constraints.

Chapter 3 will now use these ideas to develop a mathematical model to be used to solve the pilot scheduling problem.

To summarize the procedure, figure 5-2 gives the general algorithm, in flow chart form, that we will use to solve the fighter pilot scheduling problem. The following is the written form of the algorithm.

Step 0: Initialize. Read in the data and let LB^* = infinity.

Step 1: Solve (NET)-- using a min-cost network flow algorithm to obtain x^* and Z_R.

Step 2: Test the solution. Test to see if x^* is feasible with respect to the time constraints. If it is feasible or if $Z_R > LB^*$ (the best bound so far), then go to step 6. Otherwise go to step 3.

Step 3: Solve SIP_i for all i. Use an integer programming algorithm to find y^* and z_i, and therefore LB for the current candidate problem.

Step 4: Form a new problem--by changing the x variables where $y^*_{ij} = 1$ so that $x_{ij} = 0$ and $x_{rj} = 1$ (r as defined previously). If this new problem is feasible go to step 6, otherwise go to step 5.

Step 5: Select the separation variable. From the

CHAPTER 3

PROBLEM FORMULATION

The exact formulation of this problem depends on how we wish to solve it. This chapter formulates the problem as an assignment problem, assigning pilot to duties at a specified cost, with additional constraints modeling crew rest requirements and preventing a pilot from being scheduled for two duties at once.

The mathematical programming portion of the model will deal only with scheduling flights. Most of the jobs are flights, and by simplifying the problem in this manner we keep it from becoming too complicated for small computers. The computer will still aid in manual scheduling of the other duties not dealt with by the mathematical programming routine.

After we find a solution to the flying problem, the computer will display who is available for the other duties. The scheduler can then select someone to fill the duty. If there is no one available for a duty, the scheduler can assign someone, and then resolve the flight problem with that pilot now unavailable during his assigned duty.

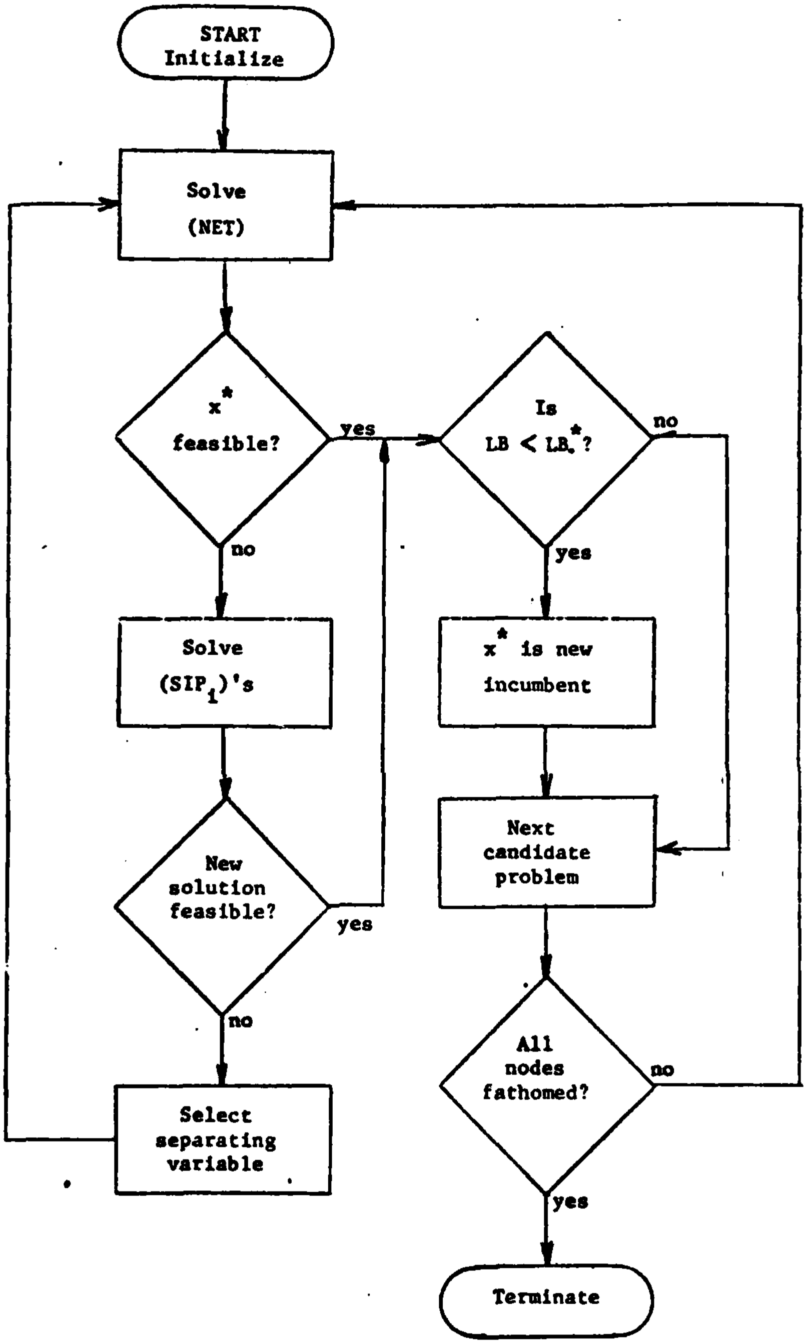


Figure 5-2

Branch and Bound Flow Chart

variables where $y^*_{ij} = 0$ select the one with the maximum p_j. Set $x_{ij} = 1$ and go to step 1.

Step 6: Test for optimality. If $LB < LB^*$ then the current solution becomes the new incumbent solution, and let $LB^* = LB$. Go to step 7.

Step 7: Select the next candidate problem. Let the last separation variable (x_{ij}) equal 0, and go to step 1. If there are no more candidate problems, terminate.

This method can be interpreted as Lagrangian relaxation, as the optimal shadow prices, v^* and w^*, from (NET) which determine the reduced costs, c_{ij}, can be viewed as the Lagrange multipliers.

5.3.4 Branch and Bound--Example

We will illustrate the procedure with a simplified example. We consider the example posed in chapter 3, except to help simplify the discussion, we will only use the first four flights (requiring 6 pilots [figure 5-3a]). We assume we have four pilots available, and can model the situation by the network in figure 5-3b. Each pilot must fly at least once, but no more than three times. Figure 5-3c specifies

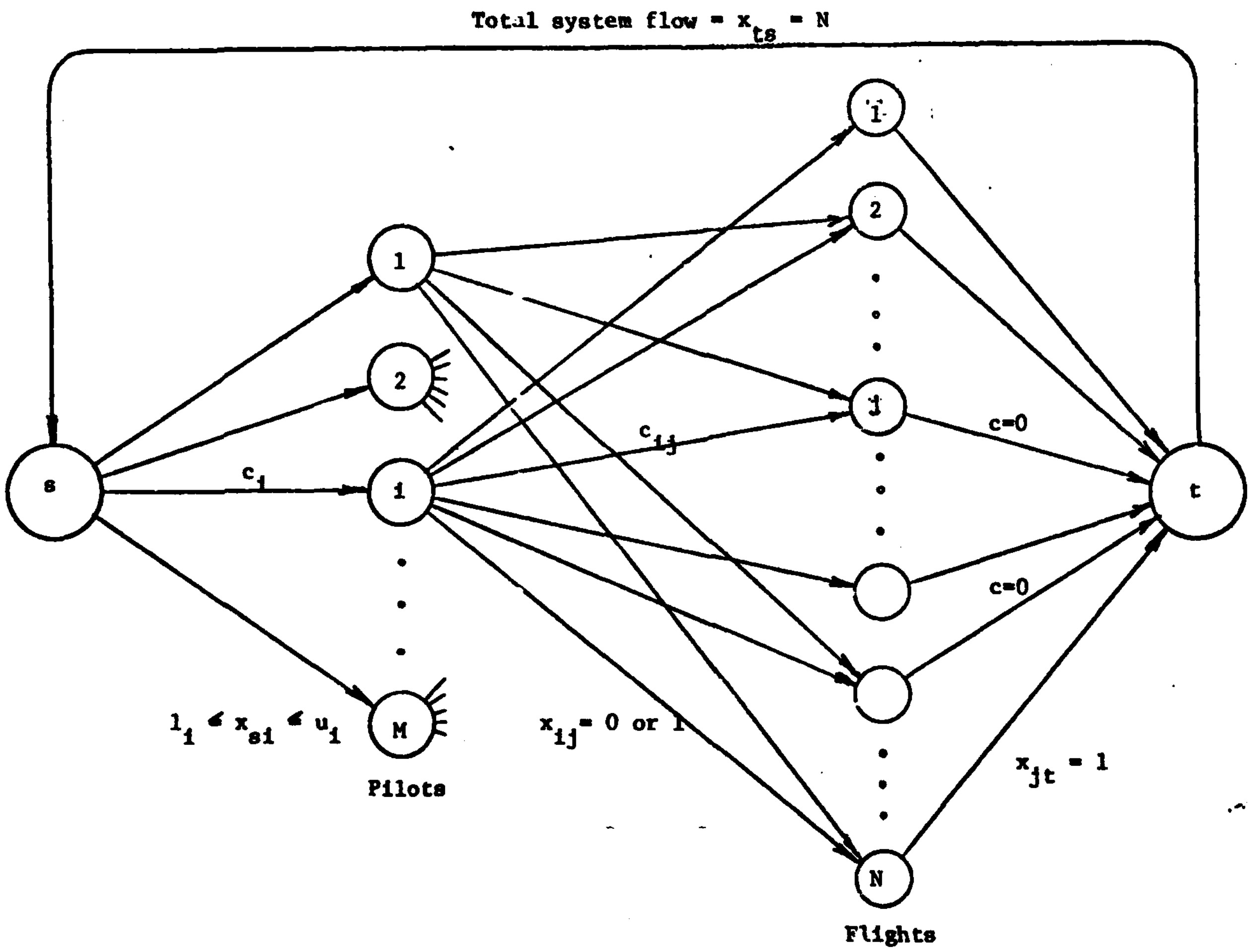


Figure 3-1
Assignment Network

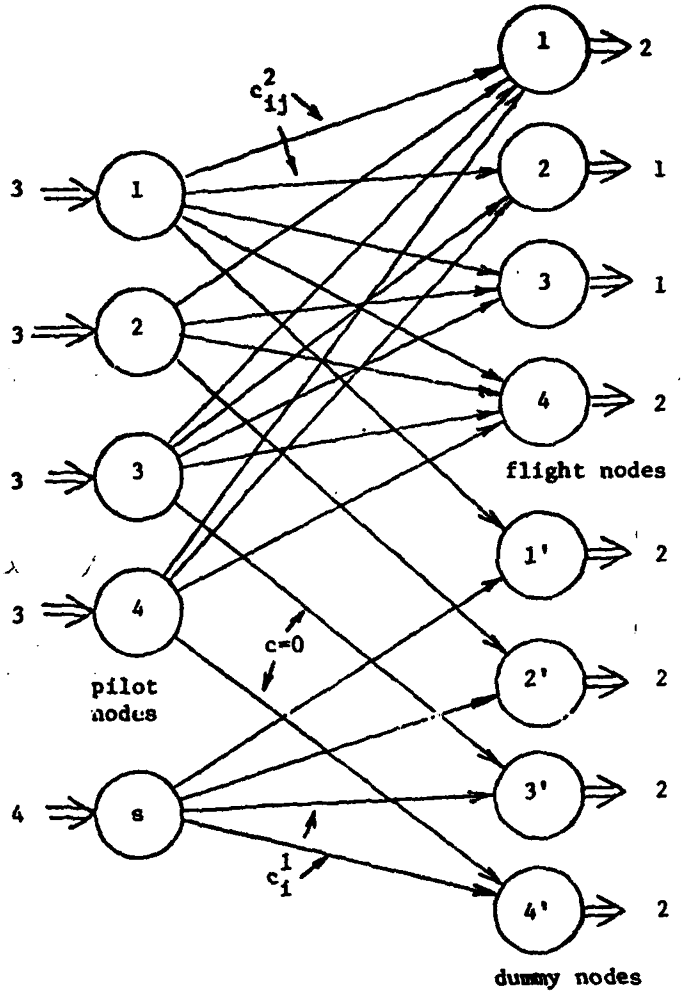


Figure 5-3a

Network Representation of the Sample Problem

$$x_{ij} = \begin{cases} 1 & \text{if pilot i is assigned flight j} \\ 0 & \text{otherwise,} \end{cases}$$

x_{si} = the total number of flights assigned to pilot i (i.e. the flow from the super source, s, to i in the network),

and let

u_i and l_i denote the upper and lower bounds on the number of flights per week for pilot i to fly.

Let us also define

$$g_{ij} = \begin{cases} 1 & \text{if pilot i is qualified for flight j} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$q_{ij} = \begin{cases} 1 & \text{if pilot i is available for flight j} \\ 0 & \text{otherwise.} \end{cases}$$

We also let

$a_{ij} = g_{ij} \cdot q_{ij}$, so that

$$a_{ij} = \begin{cases} 1 & \text{if arc i-j is feasible} \\ 0 & \text{otherwise.} \end{cases}$$

Until we define the cost function later in the chapter, we will assume a general cost function, f(x).

Day 1	Flight 1	Flight 2	Flight 3
Brief time	0515	0930	1400
Takeoff time	0715	1130	1600
Type flight	Air Combat	DART	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0830	1245	1715
End debrief time	1015	1430	1900

Day 2	Flight 4
Brief time	0500
Takeoff time	0700
Type flight	Air Refuel
	2 pilots required
Land time	0815
End debrief time	1000

Figure 5-3b
Example Problem Schedule

The assignment problem can be written

$$Z = \min f(x)$$

subject to

$$\sum_i a_{ij} x_{ij} = 1 \qquad j = 1, 2, \ldots, N \qquad (3.1)$$

$$l_i \leq x_{si} \leq u_i \qquad i = 1, 2, \ldots, M \qquad (3.2)$$

$$x_{ij} = 0 \text{ or } 1, \; x_{si} \text{ integer.} \qquad (3.3)$$

Constraints (3.1) require every flight j to have one pilot. Constraints (3.2) limit the total number of flights during the week for each pilot i.

Instead of using a formulation like this where each node j represents one flight, we can reduce the number of j nodes and therefore the problem size. For example, suppose we have 2 flight and 2 wingman slots for each flight of four aircraft to be scheduled. We can aggregate two identical nodes (i.e. flights with identical takeoff times, flight durations, pilot qualification requirements, and types), and make a new node with a demand of $b_j = 2$. The effect of this adjustment will be to decrease the number of constraints in (3.1). Equation (3.2) will remain the same. In the schedule, depicted in figure 2-2, this procedure reduces the number of flight nodes

Flights

	c_{ij}	1	2	3	4
	1	2	4	1	3
Pilots	2	3	☒	2	2
	3	1	3	2	2
	4	2	1	☒	2

Example Problem Costs (from chapter 3)

Flight

j

	f_{kj}	1	2	3	4	
	1	1	1	1		≤ 1
Constraint k	2		1	1		≤ 1
	3			1	1	≤ 1

Time Constraint Matrix for Example Problem

Figure 5-3c
Example Problem Cost and Time Constraint Matrices

from 130 to 80, a 38 percent reduction. Notice that all we have to do is change equation (3.1) to

$$\sum_i a_{ij} x_{ij} = b_j \qquad j = 1, 2, \ldots, N. \quad (3.4)$$

The decrease in problem size would help reduce the work involved in generating the cost function, and the arc-node incidence matrix, but constraint (3.1) implies the upper bound of 1 on the arc i-j, so it will be more beneficial in the solution algorithm.

3.1.2 Eliminating the Supply Bounds

Depending on the algorithm or computer code used to solve the problem, it may be useful to have a non-varying supply at the pilot nodes, instead of the variable bounded supply in our present formulation. We can accomplish this by two well known transformations: transforming the lower bound to zero and eliminating the upper bound (Golden and Magnanti [17]).

In figure 3-1 the arcs s-i are bounded by u_i and l_i, which represent the maximum and minimum number of flights per week for pilot i. To transform the lower bounds to zero, we substitute the cost (c_{ij}) and time overlap (f_{kj}) matricies, that we developed in chapter 3. An "X" in the cost matrix means that the pilot cannot fly that flight (due to other obligations).

Step 0: Initialize. LB^{*} = infinity.

Step 1: The optimal solution is the set of pairings shown circled in figure 5-4a. $Z_R = 9$.

Step 2: Pilot 4's schedule is infeasible since he is to fly both flights 1 and 2, so we go to step 3.

Step 3: We find the p_j's by looking at figure 5-4a and noting that to reassign flight 1 from pilot 4 to pilot 1 would cost nothing, and to reassign flight 2 to pilot 3 would cost 2 units. We then solve SIP_4 and find $y^{*}_{41}= 1$, and $y^{*}_{42}= 0$ (figure 5-4b). LB = 9.

Step 4: The new solution, after reassigning flight 1, is still not feasible.

Step 5: We choose x_{42} as the separation variable, so we set $x_{42} = 1$, $x_{41} = 0$, (we know x_{41} cannot equal 1 in a feasible solution). Go to step 1.

$$x'_{is} = x_{is} - l_i,$$

for x_{is}, so we have the new bounds

$$0 \leq x'_{is} \leq u_i - l_i,$$

and the supply and demands are adjusted as shown in figure 3-2. For example, if the original arc s-i had a lower bound of 2, and upper bound of 5, and a flow of 4, then the new arc formed by this transformation would have a lower bound of 0, an upper bound of 3, and a flow of 2.

We now wish to eliminate the upper bound on the new arc s-i. We start (in figure 3-3) with the arc s-i already adjusted so the lower bound is zero. Then we add a dummy node, i', between nodes s and i. We associate the cost of arc s-i with the new arc s-i', and the upper bound with arc i'-i. Now we simply reverse arc i'-i (see figure 3-3) which results in a demand of $u_i - l_i$ at i', and a net supply of u_i at node i. The upper bound is now implied by the node balance constraint at node i.

The network retains its bipartite form (figure 3-4). We can still express the problem in circulation form by using a super supply node, ss, with arcs ss-i having upper and lower

		Flights			
c_{ij}		1	2	3	4
	1	2	4	①	3
Pilots	2	3	☒	2	②
	3	①	3	2	2
	4	②	①	☒	②

$Z_R = 9$

Pilot 4 is infeasible

Soution to (NET) - no restrictions

$$z_4 = \min \quad 0y_{41} + 2y_{42}$$
$$\text{subject to} \quad y_{41} + y_{42} \geq 1$$

$z_4 = 0$, $y_{41}^* = 1$, $y_{42}^* = 0$

$LB = Z_R + z_4 = 9$

Figure 5-4a

Example Problem--First Solution

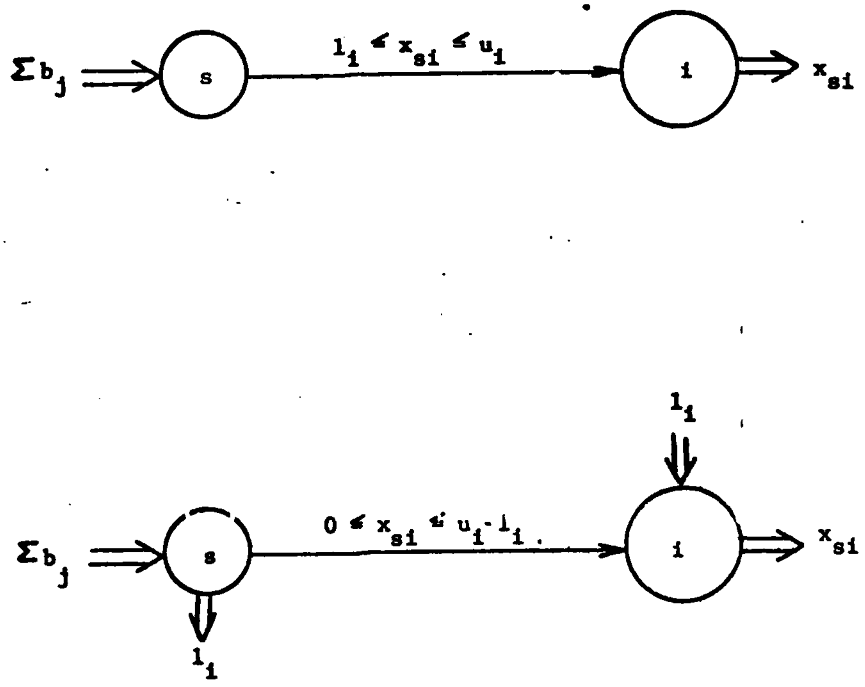


Figure 3-2
Eliminating Lower Bounds

Flights

$$\begin{array}{cc|cccc} & c_{ij} & 1 & 2 & 3 & 4 \\ \hline & 1 & \textcircled{2} & 4 & \textcircled{1} & 3 \\ & 2 & 3 & \times & 2 & \textcircled{2} \\ \text{Pilots} & 3 & \textcircled{1} & 3 & 2 & 2 \\ & 4 & \xcancel{\textcircled{2}} & \textcircled{1} & \times & \textcircled{2} \end{array}$$

LB = 9

Pilot 1 is infeasible

Figure 5-4b
Solution After First Reassignment

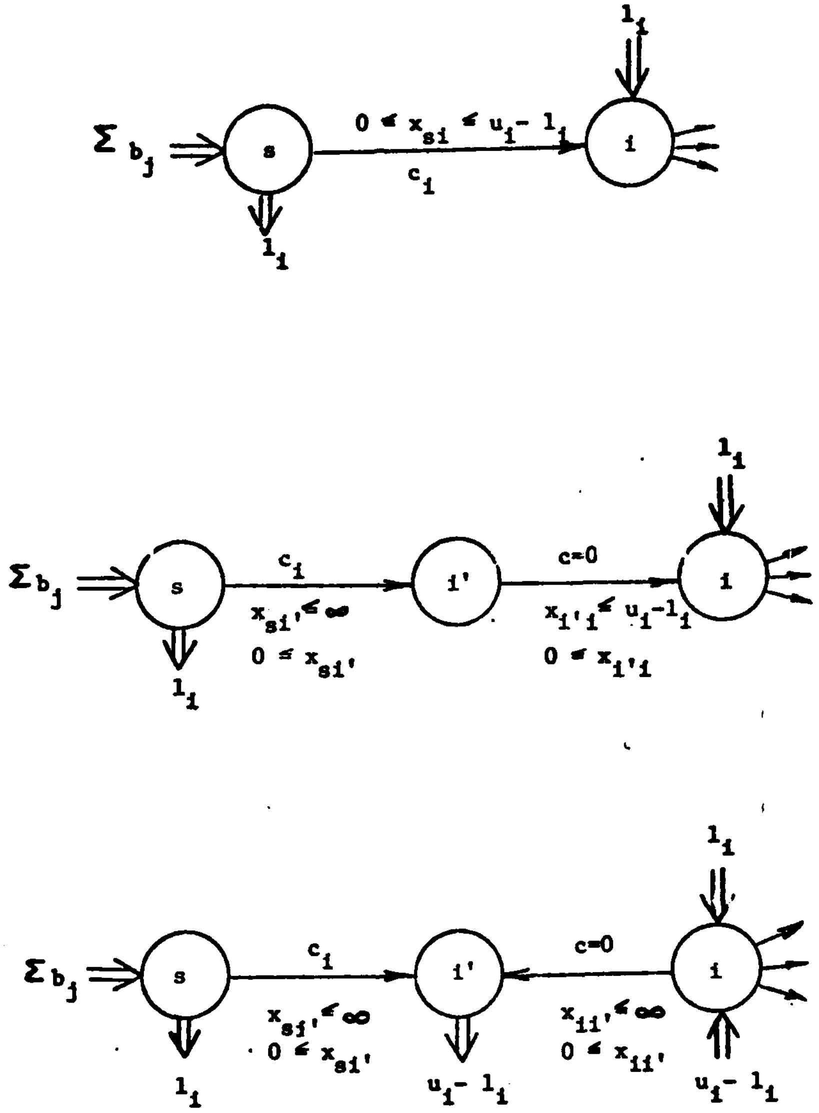


Figure 3-3
Eliminating Upper Bounds

Step 1: The solution to the candidate problem with $x_{42} = 1$ is in figure 5-5a. $Z_R = 9$.

Step 2: Pilot 1's schedule is now infeasible because he is scheduled for flights 1 and 3.

Step 3: We solve SIP_1 and find $y^*_{11} = 1$, $y^*_{13} = 0$, and LB = 10.

Step 4: Reassigning flight 1 to pilot 2 yields a feasible solution (figure 5-5b), so this candidate problem is fathomed, and we go to step 6.

Step 6: 10 is less than infinity, so $LB^* = 10$, and the candidate problem with $x_{42} = 1$ is the current incumbent solution. Step 7: We now look at the problem with $x_{42} = 0$, go to step 1.

Step 1: Figure 5-6 shows the new solution when $x_{42} = 0$.

Step 2: The optimal value is 11, which is greater than LB^*, so we go to step 6.

Step 6: The old solution is still the incumbent solution.

Flights

	c_{ij}	1	2	3	4
	1	②	4	①	3
	2	3	☒	2	②
Pilots	3	①	3	2	2
	4	☒	⓵	☒	②

$Z_R = 9$

Pilot 1 is infeasible

Solution to (NET) with $x_{42} = 1$

$$z_1 = \min \quad y_{11} + y_{13}$$
$$\text{subject to} \quad y_{11} + y_{13} \geq 1$$

$z_1 = 1$, $y^*_{11} = 1$, $y^*_{13} = 0$

LB = 10

Figure 5-5a
Example Problem—Second Solution

Flights

c_{ij}	1	2	3	4
1	☒②	4	①	3
2	③	☒	2	②

LB = 10

Pilots

		Flights			
	c_{ij}	1	2	3	4
	1	2	4	①	3
Pilots	2	3	✕	2	②
	3	①	③	2	2
	4	②	✕	✕	②

$Z_R = 11$

$Z_R > LB^*$

so the problem is fathomed

Solution to (NET) with $x_{42} = 0$

Figure 5-6

Example Problem—Third Solution

assignment to overlapping flights.

3.2 Complicating Constraints

3.2.1 Overlap Constraints

Note that flights j are arranged in chronological order. Consider j_1 and j_2 as two different flights, in the same day (where j_1 starts before j_2). We cannot have j_1 overlap any portion of j_2 and still assign one pilot to them both. We can model this situation with the multiple choice constraint

$$x_{ij_1} + x_{ij_2} \leq 1$$

for every pilot i we might want to assign to both flights. Both variables can be zero, but only one can be non-zero and have the equation satisfied.

To extend this idea, consider any flight k. Then define

$$R_k = \{k\} \cup \{j : \text{the duration of flight } k \text{ overlaps flight } j \text{ and } k \text{ starts before } j\}.$$

For every pilot i we have a series of constraints associated with every job he can fill.

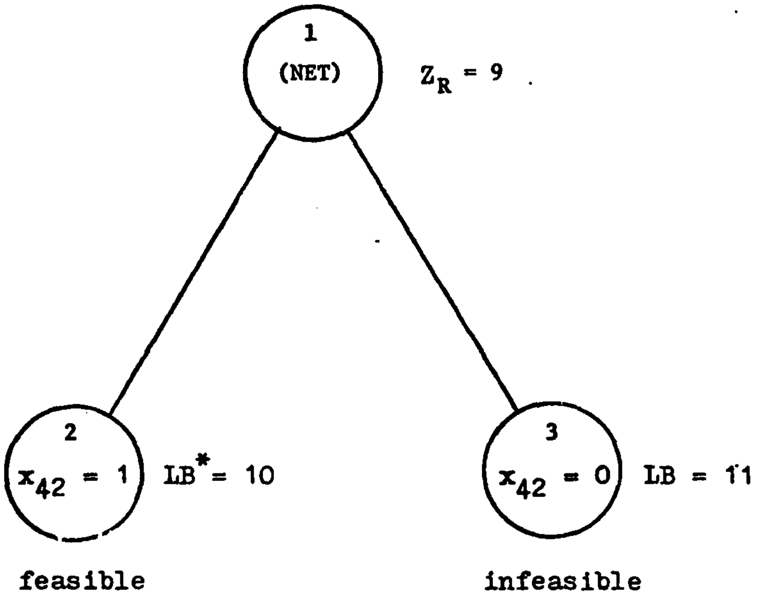


Figure 5-7
Branch and Bound Summary

$$\sum_{j \in R_K} a_{ij} \, x_{ij} \leq 1 \qquad k = 1, 2, \ldots, N. \quad (3.10)$$

The first constraint (k=1) starts with flight 1 and checks all subsequent flights (j) for time conflicts. If k and j conflict, a_{ij} is included in the summation (i.e. $a_{ij}= 1$), otherwise a_{ij} is excluded (i.e. $a_{ij}= 0$). We then add a similar constraint for flight 2 (i.e. k = 2), and so on. If flight 2 conflicts with flight 1, we do not include flight 1 in the equation k = 2. This is because the equation with k = 1 already prevents flights 1 and 2 from being scheduled at the same time. Therefore we can simplify the task of developing these overlap constraints by including only future flights in the time overlap constraint for flight k.

3.2.2 Crew Rest Constraints

For the crew duty days, we need only consider flights landing later than 12 hours after the first duty of the day. This normally means checking flights in the beginning of the day with those at the end of the day. For any flight k, we define

Step 7: There are no more candidate problems, so terminate. The optimal solution is $x_{21} = 1$, $x_{31} = 1$, $x_{42} = 1$, $x_{13} = 1$, $x_{24} = 1$, and $x_{44} = 1$, with $Z = 10$.

This example showed how we may be able to find a feasible solution by reassigning flights when $y^* = 1$, and that we can fathom candidate problems by use of the best lower bound. Figure 5-7 gives a picture of how we used the branch and bound process.

5.4 Network Problem

To find candidate solutions for x to use in the (SIP_i)'s, we must solve an assignment type min-cost network flow problem. We have three possible solution methods: the primal simplex (7), the primal-dual (5,6), and the out-of-kilter (14). See the references for explanations of the primal-dual and out-of-kilter methods.

The primal simplex method has been modified for use with min-cost network and transportation problems (17,23). The program we will use is a specialized version of the simplex method called the modified distribution method, which is used for transportation problems. Our code was adapted from Levin, Kirkpatrick, and Rubin (23), and Poole (34). The

S_k = {k} U {j: the landing time of flight j is more than
12 hours after the start of flight k, and
flight j is in the same day as flight k}.

Then to prevent someone from flying both early and late, add the multiple choice equations

$$\sum_{j \in S_k} a_{ij} \, x_{ij} \leq 1 \qquad k = 1, 2, \ldots, N \qquad (3.11)$$

to the problem for each pilot i.

Similarly the overnight crew rest requirement would only involve the late flights of one day and the early flights of the next. So if

T_k = {k} U {j: the start time of flight j is less
than 12 hours from the time at which
flight k ends},

then the associated equations for each pilot i are

$$\sum_{j \in T_k} a_{ij} \, x_{ij} \leq 1 \qquad k = 1, 2, \ldots, N. \qquad (3.12)$$

3.2.3 Reducing the Number of Complicating Constraints

algorithm finds augmenting paths at each pivot, and then pivots the new variable into the basis. We can use the "big M" method for our cost structures (i.e. infeasible pairings will have very large costs) so that we do not need to start with a feasible solution. Any solution that satisfies the supply and demand constraints (even over infeasible arcs) will serve as a starting solution. We can use the big M property to advantage during our branching process. When we set $x_{ij} = 0$ we change c_{ij} to big M and it is pivoted out of the basis. Similarly, if we wish x_{ij} to be 1, we let $c_{ij} = -M$ and x_{ij} is pivoted into the basis. We can then start the intermediate solution process from an almost feasible (and almost optimal) solution. The time required for such a solution procedure is shorter than if we solved the new problem from scratch at each iteration.

The algorithm is explained in detail in Levin, et al (23), and in many Operations Research texts. Poole (34) gives a BASIC code for the algorithm.

5.5 Time Constraint Subproblems

The final section of this chapter describes the methodology we can use to solve the subproblem (SIP_i) formulated earlier. There are two methods we will consider for possible use. The first is to convert (SIP_i) into a knapsack problem and then, using knapsack algorithms, find a solution, or second, because the problem is small, we can numbers which must be appropriately approximated to find a solution. As a result, the numbers in the problem may become very large.

Garfinkel and Nemhauser describe a method which combines constraints in pairs until all are combined into one constraint. Suppose we want to combine the constraints

$$\sum_{j=1}^{N} f_{1j}y_{ij} + s_1 = 1, \qquad (5.17)$$

and $$\sum_{j=1}^{N} f_{2j}y_{ij} + s_2 = 1 \qquad (5.18)$$

into one.

We first find a multiplication factor, α, for one constraint (say the first). We then multiply the other constraint by α, and then add the two constraints together. In our problem we can always weight the constraints by $\alpha = \sum f_{ikj} +1$ (refer to Garfinkel and Nemhauser). The new constraint is given by

$$\sum_{j=1}^{N} (f_{1j}+\alpha_1 f_{2j})y_{ij} + s_1 + \alpha_1 s_2 = 1 + \alpha_1. \qquad (5.19)$$

We can then combine the new equation with another equation, and repeat the process until only one constraint remains. If we had a large number of constraints, this method could

$$Z = \min f(x)$$

subject to

$$\sum_i a_{ij} x_{ij} = 1 \qquad \text{all } j \tag{3.14}$$

$$\sum_i x_{ii'} + x_{si'} = u_i - l_i \qquad \text{all } i' \tag{3.15}$$

$$\sum_{i'} x_{si'} = \sum_j b_j - \sum_i l_i \tag{3.16}$$

$$\sum_j a_{ij} x_{ij} = u_i \qquad \text{all } i \tag{3.17}$$

$$\sum_j f_{ikj} x_{ij} \leq 1 \qquad k = 1,\ldots, N, \text{ all } i \tag{3.18}$$

$$x_{ij},\ x_{ii'},\ x_{si'} \geq 0, \text{ integer.} \tag{3.19}$$

The problem has N +2M + 1 node balance constraints (where N is the number of flights and M is the number of pilots). Each pilot has N-1 overlap constraints, so in all we have M(N - 1) of these constraints. Thus, for example, in a problem with 7 pilots and 24 flights, the formulation has 39 node balance constraints, and 161 time overlap constraints.

produce some large numbers, but with our problem size the derived coefficients should not be excessively large.

Once we transform the set covering constraints to knapsack constraints we can solve the problem by efficient dynamic programming algorithms. Garfinkel and Nemhauser (16) give an algorithm that is appropriate for solving this problem.

5.5.2 Enumeration

Because of the small size of (SIP_1), enumeration might be almost as fast as using a knapsack algorithm. Even though the problem might have a large number of feasible solutions, on the average we would expect the problems to be very small, and solution times very small. We also eliminate the time required to transform the problem. Therefore we will use the enumeration technique when implementing the solution procedure.

3.3 Costs

So far all we know is that we wish to minimize some cost function having to do with the shortfall in TACM 51-50 requirements. We assume that f(x) is a linear combination of the individual costs of assigning pilots to flights. This choice is consistent with our use of the assignment model, so that each arc has a per unit cost in the objective function. This also means we can generate the arc costs independently; that is, the cost of one arc never depends on the cost of another.

Recall from chapter 2 that we must satisfy the requirements for the total number of flights, and for the number of each type of flight. To accomplish this goal we break the costs into two components. The first component is the cost associated with the amount flight j can contribute to satisfying pilot i's need for total flights. The second component is the cost associated with the amount flight j can contribute to pilot i's requirement for flights of type j.

We define the "cost" of a flight for pilot i to be proportional to the number of flights pilot i has already accomplished. In other words, costs will be defined as a function of the percentage of TACM 51-50 requirements pilot i has finished.

CHAPTER 6

CONCLUSION

6.1 Background

Our goal in this thesis has been to develop a model that would solve the fighter pilot problem on a micro-computer. We did not set out to develop a computer code that is in any sense best, or even efficient. Rather, we wished to establish the computational viability of using micro-computers and modern integer programming methods to solve scheduling applications such as the squadron pilot problem. Therefore, most of our observations are geared toward the problem structure, implementation issues, and a general evaluation of the method.

In order to ensure that the program would run on a micro- computer, we developed and tested our code on the IBM personal computer (IBM PC). Our particular computer was equipped with a FORTRAN 77 compiler that we decided to use for this project. The IBM PC contained 128K of internal memory and 2-320K, 5 1/4" disk drives.

Every flight, no matter what type (there are other types of flights with no minimum requirement), will count toward a pilot's total flight requirement. This cost is constant over all flights j for a given pilot i.

To satisfy the type requirements, flight j must be the same type as the requirement in question. The second cost depends on j, as well as i. Let

c_i^1 = the cost (number of flights flown/total flight requirement) associated with total flights for pilot i,

and

c_{ij}^2 = the costs (number of type j flights flown/type j flight requirements) associated with specific types of flights.

We will associate c_i^1 with arcs s-i since they apply to all flights pilot i flies. Similarly, we associate c_{ij}^2 with arcs i-j since they depend on the type of flight j is. We can weight the components to reflect the scheduler's view of which component is more important relative to the others (i.e. we may want to emphasize the completion of air refueling requirements over air combat training missions). The objective function f(x) can now be written as

To test the program we obtained old schedules from the 27th Tactical Fighter Squadron to use as the data. We then used a subset of the data for the development and initial stages of testing. We never progressed far enough to try full size problems.

6.2 Methodology

Our approach to the problem was to solve it in 3 phases: a matrix generation phase, an optimization phase, and an output phase.

The matrix generation phase takes the raw data from user data files and converts the data into a cost matrix and a feasibility matrix (as we did in the example in Chapter 3). We puc these two matrices inco files, as inputs to the optimization phase.

We had five raw data files:

1. Pilot data -- this includes the pilot's name and qualifications data,

2. Pilot accomplishment -- this file contains the number of each type of flight a pilot has flown,

$$\sum_{i'} w^1 c^1_i \; x_{si'} + \sum_i \sum_j w^2_j c^2_{ij} \; x_{ij}, \qquad (3.20)$$

where w^1 and w^2_j are appropriate weights assigned to their respective costs. The weight w^2_j can depend on what type of flight j is.

The costs are designed to model the differences in the desirability between the pilots. The weights are designed to allow the schedulers to stress one type of flight over another. For instance, the schedulers may decide that filling the requirements for DART missions is more important than filling ACTT missions because the squadron will have no more DART missions for a month (which is often the case). By making the weight larger for the ACTT missions, relative to the DART missions, we demphasise ACTT missions relative to DART missions (since we are minimizing costs).

The problem statement becomes

$$Z = \min \quad \sum_{i'} w^1 c^1_i \; x_{si'} + \sum_i \sum_j w^2_j c^2_{ij} \; x_{ij} \qquad (3.21)$$

subject to

$$\sum_i a_{ij} \; x_{ij} = 1 \qquad \text{all } j \qquad (3.22)$$

3. Pilot availability -- this file contained information concerning when a pilot was not to be available for flying duty (day and times),

4. Requirement data -- this file stores the TACM 51-50 requirements,

5. Schedule -- this file holds the schedule we wish to fill. It includes times, type of flight, and the qualifications required to fly it.

The optimization phase solved the problem using a branch and bound algorithm as we have discussed in Chapter 5. We originally tried to use a general network simplex algorithm (the code was called NETFLO [21]) to solve the relaxed network problem. The code proved to be too large for the IBM PC when imbedded in the branch and bound code. We then decided to use a code designed to solve the classical Hitchcock transportation problem (34).

The code to solve the subproblems is an enumeration method. We first develop a matrix that indicates which pairings are infeasible, so we do not have to consider all possible solutions to the problem.

$$\text{(BIP)} \qquad \sum_i x_{ii'} + x_{si'} = u_i - l_i \qquad \text{all } i' \qquad (3.23)$$

$$\sum_{i'} x_{si'} = \sum_j b_j - \sum_i l_i \qquad (3.24)$$

$$\sum_j a_{ij} x_{ij} = u_i \qquad \text{all } i \qquad (3.25)$$

$$\sum_j f_{ikj} x_{ij} \leq 1 \qquad k = 1, \ldots, N, \text{ all } i \quad (3.26)$$

$$x_{ij}, x_{ii'}, x_{si'} \geq 0, \text{ integer.} \qquad (3.27)$$

3.4 Example Formulation

Let us illustrate the formulation with a simple example. Consider the hypothetical flight schedule shown in figure 3-5, with six formations, requiring eight pilot assignments. In the example we have four pilots available. Suppose that we require each of the pilots to fly at least one, but no more than three flights. Suppose too, that pilots 2, 3, and 4 are unavailable for flights 2, 6, and 3 respectively. The resulting node adjacency matrix $\{a_{ij}\}$ and time overlap constraint matrix $\{f_{kj}\}$ (constraints (3.26)) are shown in figure 3-6. Note that this matrix is strictly showing the conflicts between flights. We will add the restriction that a pilot i be available and qualified (i.e. $a_{ij}= 1$) at a later time.

The branch and bound code directs the program flow and keeps track of the current candidate problem. It puts bounds on the variables by changing costs depending on whether we want the variable at 1, 0, or free (e.g., cost equals "M" if the variable is restricted to zero or equals "-M" if the variable is restricted to 1).

We use a depth first search to find a feasible solution quickly. If we find a feasible solution early in the enumeration procedure, we can reduce the number of problems to be considered. We also include the option of stopping at the first feasible solution, which might be useful for problems that are too large to solve to optimality or for problems where we obtain "good" or near optimal solutions before terminating the complete branch and bound eumeration.

At each branch we use the feasibility matrix (as in the example problem) to exclude all variables that conflict with the separation variable. This hopefully helps lead to a feasible solution. If our transportation algorithm then yields a solution that includes infeasible arcs, we know there are no feasible solutions along that branch, so we can fathom the branch.

Once it has discovered the solution to the problem, the program writes it into a file for the output generation

Day 1	Flight 1	Flight 2	Flight 3
Brief time	0515	0930	1400
Takeoff time	0715	1130	1600
Type flight	Air Combat	DART	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0830	1245	1715
End debrief time	1015	1430	1900

Day 2	Flight 4	Flight 5	Flight 6
Brief time	0500	0930	1400
Takeoff time	0700	1130	1600
Type flight	Air Refuel	Air Combat	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0815	1245	1715
End debrief time	1000	1430	1900

Figure 3-5

Example Problem Schedule

phase.

The output generation phase contains a short program to sort the solution and display it in a form useful to the user.

Appendix C contains the computer code of the 3 programs.

6.3 Results

Our first concern was that the cost structure would lead to unstable solutions. Many of the flight categories have requirements for only 2 to 4 flights (e.g., DART and INST) and in our data many pilots had not accomplished any, meaning that many of the costs were essentially zero. We were concerned that this degeneracy would have a serious effect on our ability to obtain a solution.

We found, in the transportation algorithm, that 70 per cent of the pivots were degenerate, in that they involved no transfer of flow. They only moved variables in and out of the basis. The algorithm did, however, find optimal solutions each time it was used.

Flights

Pilots

i \ j	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	0	1	1	1	1
3	1	1	1	1	1	0
4	1	1	0	1	1	1

Node-node adjacency matrix

Flight

Flight

k \ j	1	2	3	4	5	6
1	1	1	1			
2		1	1			
3			1	1		
4				1	1	1
5					1	1
6						1

Feasibility constraint matrix

Figure 3-6

Example Problem Data

This means that the subproblems consumed the major share of the solution time. Reducing the solution time would require an efficient algorithm for the subproblems (such as a good 0-1 knapsack algorithm).

Another finding was that the number of pilots unavailable to fly due to other commitments had a significant impact on the ability to find a feasible solution (to BIP). Problems with relatively few instances of unavailable pilots were solved much faster than problems where pilots had numerous other duties.

The internal memory of the IBM PC is capable of handling our program and data. The storage required for an 8 by 25 problem is only 6.5K. The execution code requires 56K of storage.

6.4 Conclusion

The methods we have discussed do solve the fighter pilot scheduling problem. There is, however, room for improvement. The computer code could be improved to accelerate computations. There may be better algorithms (such as the more complicated multiplier adjustment method) to solve the problem. In the future, we hope to see if any of these

		Total	ACTT	DART	NINT	AARD
	weight	1	1	1	1	1
	1	1	1	2	0	2
Pilot	2	2	1	N/A	1	0
	3	2	0	1	0	0
	4	1	0	0	0	1

Completion percentages
(in decimal form [i.e. 1 = 100%])

note: a large weight deemphasizes the type of flight, here we weight all types evenly

Flight

	j i	1 ACTT	2 DART	3 NINT	4 AARD	5 ACTT	6 NINT
	1	2	3	1	?	2	1
Pilot	2	3	☒	3	2	3	3
	3	2	3	2	2	2	☒
	4	1	1	☒	2	1	1

Example cost matrix

Figure 3-7
Example Problem Costs

methods can be successfully implemented on a micro-computer.

Let us analyze our program with respect to the goals we set for ourselves in Chapter 2. The first goal is to ensure that TACM 51-50 flight requirements are met. We accomplish this through our objective function. Our costs are such that, those pilots who are behind relative to other pilots will be scheduled more often. Although this approach does not ensure all flight requirements will be met, it does tend to keep anyone from lagging behind. Moreover, it gives the schedulers the flexibility to change scheduling priorities for the pilots by changing the cost structure.

The second goal is to ensure that each pilot's minimum and maximum number of flights per week are observed. Our transportation algorithm, by virtue of our lower and upper bound transformations ensures that we comply with this restriction.

The third goal is to ensure no pilot flies without proper rest, flies with too long a duty day, or is scheduled when not available to fly. Our development of the overlap constraints and the feasibility matrix ensure that no one is scheduled during those times.

Pilot 1							Pilot 2						Pilot 3						Pilot 4						s				
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	$x_{11'}$	x_{21}	x_{23}	x_{24}	x_{25}	x_{26}	$x_{22'}$	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	$x_{33'}$	x_{41}	x_{42}	x_{44}	x_{45}	x_{46}	$x_{44'}$	$x_{s1'}$	$x_{s2'}$	$x_{s3'}$	$x_{s4'}$	RHS
1	3	0	2	1	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	2	2	1	Z
1							1						1						1										2
	1													1						1									1
		1						1							1														1
			1						1							1					1								2
				1						1							1					1							1
					1						1												1						1
						1																			1				2
												1														1			2
																		1									1		2
																								1				1	2
																									1	1	1	1	4
1	1	1	1	1	1	1																							3
							1	1	1	1	1	1																	3
													1	1	1	1	1	1											3
																			1	1	1	1	1	1					3
1	1	1																											1
	1	1																											1
		1	1																										1
			1	1	1																								1
				1	1																								1
							1	1																					1
								1	1																				1
									1	1	1																		1
										1	1																		1
													1	1	1														1
														1	1														1
															1	1													1
																1	1												1
																			1	1									1
																					1	1	1						1
																						1	1						1

Figure 3-8

BIP Formulation of the Example Problem

The fourth objective is to solve the problem in less time than the present system. The present system takes about two man-days of work to find a "good" schedule. Once proficient wich the data structures, schedulers could solve the problem in less than 1 hour, including inputting data into the data files and running the program. Clearly, using this program would provide time savings for the schedulers and free them for other tasks.

The fifth goal is to run the program on a micro-computer. We have successfully accomplished this, however, we have not tried full-scale problems yet. The storage requirements for our sample problems were well within the capabilities of the IBM PC, and we postulate that we could, in fact, solve problems of 30 pilots and 120 flights on this computer.

We did well on the five goals we stated, but we also mentioned that we would like to have auxiliary programs that are useful in daily decision making. We were not successful on this point as time did not permit us to concentrate on that aspect of the model. In addition to efforts in bettering the optimization code, we would like to see someone develop a user friendly interface with the program, so that non-technical people could effectively run the optimization.

To help understand f_{kj} consider row 1 in $\{f_{kj}\}$. The first three 1's mean that flight 1 conflicts with flights 2 and 3. Row 3 shows that flights 3 and 4 conflict (because of overnight crew rest), and row 4 shows that flights 4, 5, and 6 conflict.

To develop the cost matrix $\{c_{ij}\}$ we assign weights, as shown in figure 3-7, to the cost components, and multiply them by the hypothetical completion percentages (also in figure 3-7). The resulting cost matrix is the last matrix depicted in figure 3-7.

As an example, consider pilot 1 and flight 2. The cost c_{12}, is the weight for total flights (1), times the completion percentage of total flights for pilot 1 (100 per cent = 1), plus the weight for DART missions (1), times the completion percentage (2), which is $1 \cdot 1 + 1 \cdot 2 = 3$. So $c_{12} = 3$, as shown in figure 3-7.

Figure 3-8 shows our sample problem, expanded in the form of (BIP). The first 6 constraints are the node balance constraints for the flights. The second 4 constraints are for the i' nodes. The next 5 constraints are for s and the pilots. The last 16 constraints are from the $\{f_{jk}\}$ matrix, but are now adjusted for the individual pilots. We will use a portion of this problem to illustrate the solution

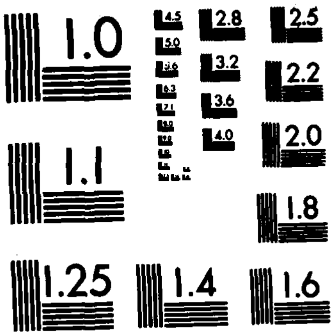
AD-A126 947 PILOT SCHEDULING IN A FIGHTER SQUADRON(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH W H ROEGE FEB 83 AFIT/CI/NR-83-2T 2/2

UNCLASSIFIED F/G 5/9 NL

END

FILMED

procedure in chapter 5.



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CHAPTER 4

REVIEW OF THE LITERATURE

Scheduling has many applications. One major application, job shop and machine job scheduling problems (29), have been studied for many years. Conway, Maxwell, and Miller (9) is a general reference to these problems. The airline crew scheduling problem (1,30,32,36) has also received much attention in the literature. Vehicle delivery problems (4) (as opposed to routing problems) have also been studied by many researchers. Scheduling algorithms also apply to staffing problems, such as the nurse scheduling problem (2,28). Miller (27) gives a survey of personel scheduling methods as they apply to the public sector.

In general, a personel scheduling problem models situations in which persons are to be assigned to a subset of jobs based on some criteria. This chapter will review the literature dealing with a particular class of applications, airline pilot scheduling, and with procedures applicable to the fighter pilot scheduling problem.

4.1 Airline Crew Scheduling

We are convinced that the use of Operations Research and Computer Science planning tools, such as those discussed in this thesis, are of great benefit to the Air Force. Specifically, we believe that these tools can be used at the Squadron and Wing levels, not only for pilot scheduling, but for many of a number of similar scheduling and allocation problems.

Arabeyre, Fearnley, Steiger, and Teather (1) survey the early attempts to solve the airline pilot scheduling problem. Most researchers separated the problem into two parts, (1) assigning flight legs (one takeoff to one landing) to rotations (a round trip of one to three days), and (2) assigning pilots to the rotations. The first problem attempted to minimize "dollar" costs, such as costs of overnight lodging. The second problem aimed to distribute pilot monthly flight time evenly.

Usually researchers and practitioners considered the first problem to be the most difficult since it had to deal with complicating constraints due to union rules, FAA regulations, and company policies. Etcheberry (11) developed an implicit enumeration algorithm, using a branch and bound framework with Lagrangian relaxation, to solve large set covering problems such as this one.

Rubin (36) solved the problem by reducing the number of constraints as much as possible before solving it. He would then consider subsets of the constraint matrix columns, find the best solution over that subset, and repeat the process until obtaining a satisfactory solution.

Marsten (26) developed an algorithm to solve the related set partitioning problem. This algorithm ordered the

APPENDIX A

FLIGHT TYPES

Air Combat Training (ACTT).

These are missions where similar types of aircraft practice "dogfight" maneuvers against each other. Weapons launches and weapons parameters are simulated and evaluated with gun camera film (42 of these flights are required every 6 months).

Dissimilar Air Combat Training (DACT).

These missions are the same as ACTT, except they are flown against other types of aircraft (DACT flights are included in the ACTT requirements).

Airborn Gunnery Practice (DART).

This mission involves firing the 20MM cannon at a metal target (Dart) which is towed 1500 feet behind another aircraft (1 or 2 of these missions are required depending on the pilot's experience level).

Intercept Training (DINT).

Intercept training involves using electronic means (e.g. RADAR) to find and simulate firing on a target. Maneuvers are much more restricted than in ACTT or DACT due to the limitations of the equipment (5 or 6 of these missions are required depending on the pilot's experience level).

Night Intercept Training (NINT).

Night intercepts are the same as day intercepts, except they must be performed at night (4 are required per 6 month period).

Air to Air Refueling (AARD).

A specially modified Boeing 707 or DC-10 carries fuel and the fighters practice intercepting the "tanker" and taking on gas through an 18 foot long "boom" on the tail end of the tanker (2 required).

constraint matrix lexicographically before starting the optimization. The algorithm then takes advantage of the constraint structure to help fathom candidate problems quickly.

Garfinkel and Nemhauser (15) developed a set-partitioning procedure that reduces the problem size by eliminating row and column vectors before applying their algorithm. The algorithm then orders the data so the rows with the least number of non-zero entries appear first, and the columns with the lowest costs are on the left. They then use an implicit enumeration algorithm that takes advantage of this structure to build possible solutions. Pierce (33) independently developed a similar algorithm.

Nicoletti (32) viewed the second problem (assignment of pilots to rotations) as a network assignment problem and successfully used the out-of-kilter method to find solutions.

The fighter pilot scheduling problem differs from the airline scheduling problem in that all the fighter flights originate and terminate at the same base. This eliminates the need to develop rotations, although we still must deal with crew rest and other regulations, just as the airlines must.

Night Air to Air Refueling (NAAR).

Night air to air refueling is the same as day refueling except that it must be accomplished at night (1 required).

Instrument Proficiency Flights (INST).

These flights are dedicated to practicing instrument approaches and other instrument procedures. The are only required for non-experienced pilots (2 every 6 months).

A possible formulation of the fighter pilot scheduling problem is in the form of "k-duty period" scheduling problem. This problem deals with schedules consisting of k independent contiguous scheduling periods; for example, a schedule might assign a person to work k four hour shifts each separated by two hour breaks.

Shepardson (40) deals with this problem. The general idea is to start with a proposed (yet feasible) subset of schedules as the columns of a constraint matrix, with its rows being the jobs to be filled. He then separates the columns into new columns each with only one contiguous scheduling period. For example, he would separate a 2-duty schedule containing two 4-hour shifts into two columns each representing a single 4-hour shift. This solution strategy is attractive because problems in which all schedules have a single contiguous duty can be solved as network flow problems.

He then adds extra side equations to ensure that if a new column is in the solution, then all the new columns associated with its column in the original problem formulation, are also in the solution. In our example, if one of the two columns with the 4-hour shifts is in the solution, then they both must be in the solution. He then dualizes these side equations and uses Lagrangian relaxation

APPENDIX B

ADDITIONAL DUTIES

Supervisor of Flying (SOF).

Only Lt Colonels, Majors, and very senior Captians who are experienced pilots may serve as SOF. The SOF sits in the control tower, and is responsible for the entire flying operations of the Wing. He has the authority to cancel flights due to weather or other circumstances. He also is there to assist any aircraft in time of an emergency, since he can call on other aircraft, fire trucks, and other resources for help.

Runway Supervisory Officer (RSO).

All MR pilots are qualified to serve as RSO. SOF's are qualified, but do not serve as RSO. The RSO serves in a special building near the end of the runway. He ensures the landing patterns are safe and that everyone lands with their landing gear down. He can also assist in emergencies by looking over the emergency aircraft for obvious exterior problems when it flies by.

Range Training Officer (RTO).

RTO's must be MR and have some experience. Approximately half the pilots are qualified to be RTO's. The RTO monitors flights which fly on a range where ground stations receive flight information from aircraft and feed the information into a computer. The computer then displays the flight on a video screen. The RTO can see a "God's eye" view of the live action and warn pilots of any dangers. The information is stored, and can be replayed in the flight debrief. The RTO monitors the live flight for safety, simulates missle launches in the computer, and relates the missle results to the fliers.

methods to solve the problem.

4.2 Lagrangian Relaxation

In problems with many complicating constraints, Lagrangian relaxation techniques that exploit underlying problem structure (like the single-duty problem that can be solved as a network flow problem) have so far yielded very good results for a wide variety of applications. Fisher (12), Magnanti (25), and Shapiro (39) all give good surveys of Lagrangian relaxation methods, and mention a number of application areas.

Lagrangian relaxation methods attempt to simplify the problem by dualizing some constraints, multiplying them by Lagrange multipliers, and adding them to the objective function. Given a set of multipliers, the relatively easy relaxed problem is solved. Then given the new solution to the relaxed problem, we solve for new multipliers. (In section 5.2 we describe the multiplier selection procedure in more detail.) We can embed this method into a branch and bound framework to systematically exhaust all possibilities, and find the optimal solution. (See (7) and (12) for an explanation of branch and bound methodology.)

APPENDIX C

COMPUTER CODES

These codes were written in FORTRAN 77 for the IBM personal computer.

The first program converts the raw data from the data files into the cost and feasibility matrices.

The second program is the optimization program that takes the cost and feasibility data and outputs the optimal schedule.

The third program is a short program to format the output as an easy to read document.

Several methods have been proposed to solve for the Lagrange multipliers. The most popular method is subgradient optimization. The method starts with a proposed solution for the multipliers, then uses a subgradient of that solution to move to a better solution. Held, Wolfe, and Crowder (18) give a comprehensive explanation and evaluation of subgradient optimization. Other methods include generalized linear programming (25), the BOXSTEP method (19), dual ascent (10), and so called multiplier adjustment methods (10,13). None of these methods has performed as well as subgradient optimization so far on a wide variety of problems, though multiplier adjustment methods have proved to be successful on facility location problems (Erlenkotter [10]) and generalized assignment problems (13).

Ross and Soland (35) proposed a heuristic for finding multipliers when solving the generalized assignment problem. They relax the supply node bounds and solve the relaxed assignment problem. Their method then assigns multipliers based on the minimum penalty (increase in cost) incurred to make the relaxed solution feasible. For each supply node whose supply bound is exceeded, they find a new assignment that makes that node supply feasible with minimal cost increase. The increase in cost for that node is its new multiplier. These multiplier problems are in the form of knapsack problems (i.e. integer programs with only one

C.1 Program to Organize Raw Data into Problem Data

```
      PROGRAM FILGEN
   THIS PROGRAM TAKES THE RAW DATA FILES
     AND PROCESSES THEM TO DATA THE PILOT
     OPTIMIZATION PROGRAM CAN USE.
      INTEGER*2 FEAS(1200),P(30,2),FPOINT(150),C(30,150),
     *ACC(30,9),AVL(30,10,4),REQ(3,9),S(150,4),SCH(150,3),
     *NE(30),ENDDAY(5),NF,NPIL,NFLT,I,J,K,UL,J1,MAX,SLI
      INTEGER*4 BIG
      CHARACTER*4 PC(30,2),T(150,2)
      DATA BIG/3200/

   OPEN THE DATA FILES

      OPEN(1,FILE='PILOT.DAT',STATUS='OLD')
      OPEN(2,FILE='ACCOMP.DAT',STATUS='OLD')
      OPEN(3,FILE='AVAIL.DAT',STATUS='OLD')
      OPEN(4,FILE='REQMNT.DAT',STATUS='OLD')
      OPEN(5,FILE='SCHED.DAT',STATUS='OLD')
      OPEN(6,FILE='COST.DAT',STATUS='NEW')
      OPEN(7,FILE='FEAS.DAT',STATUS='NEW')

   READ INTO THE PROGRAM THE RAW DATA FILES

      READ(1,1000) NPIL
 1000 FORMAT(//I5)
      DO 5 I=1,NPIL
      READ(1,1010) (P(I,J),J=1,2),(PC(I,J),J=1,2)
 1010 FORMAT(10X,2I5,3X,A2,4X,A1)
    5 CONTINUE

      READ(2,1020) (ACC(1,J),J=1,9)
 1020 FORMAT(//10X,9I5)
      DO 6 I=2,NPIL
      READ(2,1025) (ACC(I,J),J=1,9)
 1025 FORMAT(10X,9I5)
    6 CONTINUE

      READ(3,1030) NE(1)
 1030 FORMAT(//10X,I5)
      IF(NE(1).EQ.0) GOTO 8
      DO 7 J=1,NE(1)
      READ(3,1035) (AVL(1,J,K),K=1,4)
 1035 FORMAT(15X,I3,I7,I3,I7)
    7 CONTINUE
    8 CONTINUE
```

constraint). Until multiplier adjustment methods were developed, their method seemed to be faster than any other for solving generalized assignment problems, their advantage

```
      DO 10 I=2,NPIL
      READ(3,'(10X,I5)') NE(I)
      IF(NE(I).EQ.0) GOTO 10
      DO 9 J=1,NE(I)
      READ(3,1035) (AVL(I,J,K),K=1,4)
    9 CONTINUE
   10 CONTINUE

      READ(4,1050) (REQ(1,J),J=1,9)
 1050 FORMAT(//10X,9I5)
      DO 20 I=2,3
      READ(4,1055) (REQ(I,J),J=1,9)
 1055 FORMAT(10X,9I5)
   20 CONTINUE

      READ(5,1060) NFLT
 1060 FORMAT(//I5)

      READ(5,1065) (ENDDAY(I),I=1,5)
 1065 FORMAT(5I5)

      DO 50 I=1,NFLT
      READ(5,1070) (T(I,J),J=1,2),(S(I,J),J=1,4)
 1070 FORMAT(6X,A4,3X,A2,I5,I3,I2,I5)
   50 CONTINUE

  END OF READING PORTION OF THE PROGRAM

  MAIN BODY OF THE PROGRAM

      WRITE(6,1100) NPIL,NFLT
 1100 FORMAT(1X,2I5)

      SLI=0
      DO 65 I=1,NPIL
      SLI=SLI+P(I,2)
      UL=P(I,1)-P(I,2)
      WRITE(6,1110) P(I,1),UL
 1110 FORMAT(1X,2I5)
   65 CONTINUE
      WRITE(6,1110) NFLT-SLI,NFLT-SLI


      CALL ARCMAT(NFLT,NPIL,ACC,AVL,REQ,PC,T,
     *NE,SCH,S,P,C)

      DO 70 J=1,NFLT+NPIL
      WRITE(6,1115) (C(I,J),I=1,NPIL+1)
 1115 FORMAT(1X,8I5)
   70 CONTINUE
```

```
      DEVELOP THE FEASIBILITY MATRIX

      NF=0
      DO 130 J=1,NFLT
      FPOINT(J)=NF+1
      MAX=J+30
      IF(MAX.GT.NFLT) MAX=NFLT
      DO 90 K=J,MAX
      IF(SCH(J,3).GE.SCH(K,1)) THEN
      NF=NF+1
      FEAS(NF)=K
      ELSE
      K=MAX
      ENDIF
   90 CONTINUE
      CREW DUTY DAYS
      J1=ENDDAY(S(J,4))
      DO 100 K=J1-12,J1
      IF ((SCH(J,1)+1200).LT.SCH(K,2)) THEN
      NF=NF+1
      FEAS(NF)=K
      ENDIF
  100 CONTINUE
      CREW NIGHTS
      IF(S(J,4).EQ.4) GOTO 130
      DO 110 K=J1+1,J1+13
```

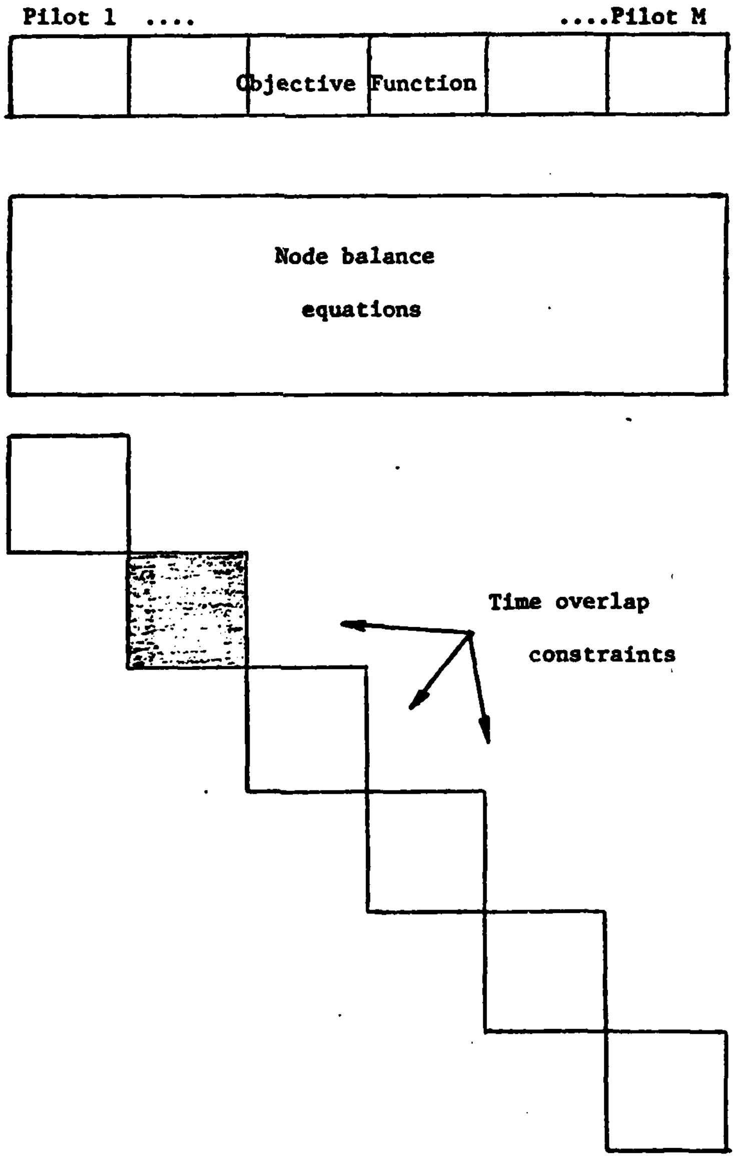


Figure 5-1a
Problem Structure

```
 THIS SUBROUTINE DEVELOPS THE ARC MATRIX

    SUBROUTINE ARCMAT(NFLT,NPIL,ACC,AVL,REQ,PC,
   *T,NE,SCH,S,P,C)
    INTEGER*2 NFLT,NPIL,ACC(30,1),C(30,1)
    INTEGER*2 AVL(30,10,1),REQ(3,1),NE(1)
    INTEGER*2 ED,U,S(150,1),SCH(150,1),P(30,1)
    CHARACTER*4 PC(30,1),T(150,1)
    INTEGER*2 DAY1,DAY2,I,J,K1,T1,T2,BTIME,ETIME
    INTEGER*4 BIG
    DATA BIG/3200/

    DO 150 I=1,NPIL+1
    DO 140 J=1,NFLT+NPIL
    C(I,J)=3200
140 CONTINUE
150 CONTINUE

    DO 250 J=1,NFLT
    DAY1=(S(J,4)-1)*2400
    SCH(J,1)=((S(J,2)-2)*100)+DAY1+S(J,3)
    SCH(J,3)=((S(J,2)+3)*100)+DAY1+S(J,3)
    SCH(J,2)=((S(J,2)+1)*100)+DAY1
    ED=S(J,3)+30
    IF(ED .GE. 60) THEN
    ED=ED-60
    SCH(J,2)=SCH(J,2)+100
    ENDIF
    SCH(J,2)=SCH(J,2)+ED

    IF(T(J,1).EQ.'ACTT') THEN
    T1=2
    ELSEIF(T(J,1).EQ.'DACT') THEN
    T1=3
    ELSEIF(T(J,1).EQ.'DART') THEN
    T1=4
    ELSEIF(T(J,1).EQ.'NINT') THEN
    T1=5
    ELSEIF(T(J,1).EQ.'DINT') THEN
    T1=6
    ELSEIF(T(J,1).EQ.'INST') THEN
    T1=7
    ELSEIF(T(J,1).EQ.'AARD') THEN
    T1=8
    ELSE
    T1=9
    ENDIF
```

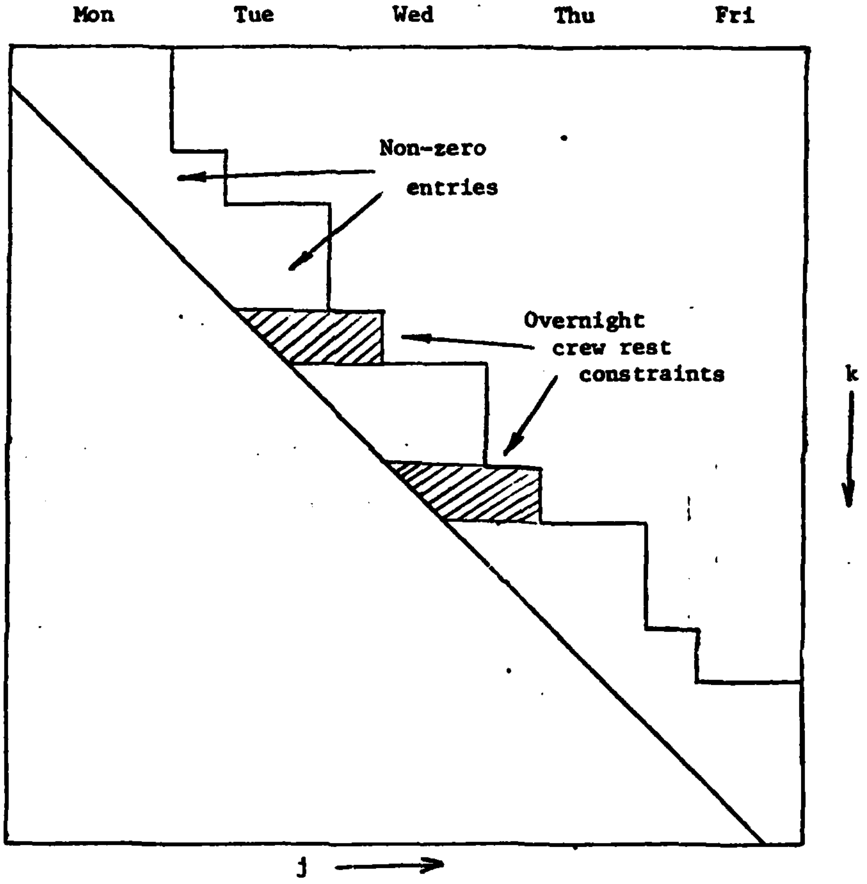
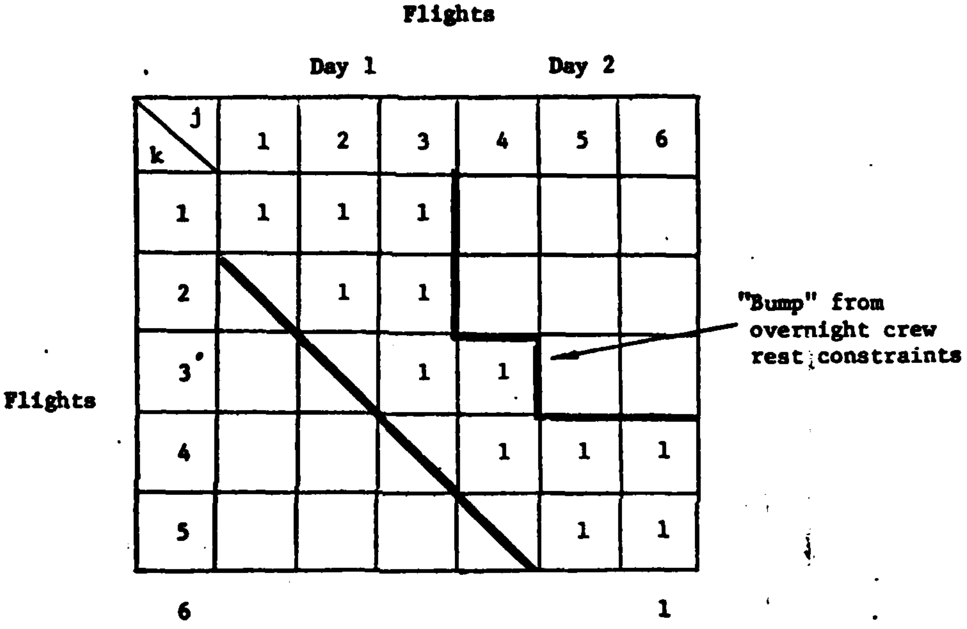


Figure 5-1b
Time Overlap Constraint Structure

```
      DO 230 I=1,NPIL
      U=1
      IF ((PC(I,1).EQ.'WG') .AND. (T(J,2) .EQ. 'FL')) GOTO 230
      DO 200 K1=1,NE(I)
      DAY2=(AVL(I,K1,1)-1)*2400
      BTIME=DAY2+AVL(I,K1,2)
      ETIME=((AVL(I,K1,3)-1)*2400)+AVL(I,K1,4)
      IF ((ETIME.GT.SCH(J,1)).AND.(BTIME.LT.SCH(J,3))) THEN
      U=0
      K1=NE(I)
      ENDIF
  200 CONTINUE

      IF (U .EQ. 1) THEN
      IF (PC(I,2).EQ.'N') THEN
      T2=1
      ELSEIF (PC(I,2).EQ.'E') THEN
      T2=2
      ELSE
      T2=3
      ENDIF
      IF((REQ(T2,T1).EQ.0).AND.(T2.NE.3)) THEN
      C(I,J)=BIG
      ELSEIF((REQ(T2,T1).EQ.0).AND.(T2.EQ.3)) THEN
      C(I,J)=(3*(ACC(I,1)*100)/REQ(T2,1))+5
      ELSE
      C(I,J)=((ACC(I,T1)*100)/REQ(T2,T1))+5
      ENDIF
      IF((PC(I,1).EQ.'FL').AND.(T(J,2).EQ.'WG'))
     *    C(I,J)=C(I,J)*2
      ENDIF
  230 CONTINUE
  250 CONTINUE

      DO 260 I=1,NPIL
         C(NPIL+1,NFLT+I)=((ACC(I,1)*100)/REQ(T2,1))+5
         C(I,NFLT+I)=0
  260 CONTINUE
      RETURN
      END
```



k \ j	1	2	3	4	5	6
1	1	1	1			
2		1	1			
3			1	1		
4				1	1	1
5					1	1
6						1

"Bump" from overnight crew rest constraints

Figure 5-1c
Feasibility Constraint Matrix

```
      DAY=0
      N=1
   40 CONTINUE
      DAY=DAY+1
      WRITE(4,1100) DAY
 1100 FORMAT('1','DAY ',I2)
      PER=0
      FLT=0
   50 CONTINUE
      PER=PER+1
      WRITE(4,1110) PER
 1110 FORMAT('0','  PERIOD ',I2)
   60 CONTINUE
      FLT=FLT+1
      WRITE(4,1120) FLT
 1120 FORMAT('0','FLIGHT',I2)
      WRITE(4,1130) TYPE(N)
 1130 FORMAT('+',2X,A5)
      WRITE(4,1140) FLTN(N,2)
 1140 FORMAT('+',2X,I5)
   70 CONTINUE
      WRITE(4,1150) PNAME(X(N))
 1150 FORMAT('+',2X,A10)
      IF(N.EQ.NFLT) GOTO 80
      N=N+1
      IF(FLTN(N-1,3).EQ.FLTN(N,3)) GOTO 50
      IF(FLTN(N-1,4).EQ.FLTN(N,4)) GOTO 60
      IF(FLTN(N-1,5).EQ.FLTN(N,5)) GOTO 70
      GOTO 40
   80 CONTINUE
      STOP
      END
```

line and the staircase within the matrix. The shaded "bumps" represent the crew rest constraints that link one day's schedule to the next. If the overnight crew rest constraints weren't present, the subproblems would separate further into daily subproblems. For example, figure 5-1c shows the time constraints for one pilot in the example problem developed in section 5.3. The arrow shows the "bump" resulting from the overnight crew rest constraint. If flights 3 and 4 didn't conflict, then the constraints for day 1 and day 2 would be separable.

5.2 Lagrangian Relaxation

We could conceivably attempt to use general purpose integer programming algorithms to solve this problem, but because of the complexity of the time constraints, these methods probably would not be very efficient. This brute force approach does not take advantage of the network structure in the common constraints, which we can exploit to solve the problem much more efficiently. By using a Lagrangian relaxation algorithm, we can take advantage of the network structure and decrease our solution times.

Fisher (12), Magnanti (25), and Shapiro (39) give a good description of the Lagrangian technique and give many

C.2 Optimization Program

```
     PROGRAM SOLUTN

     INTEGER*4 LB,LBSTAR,BIG,NEG,C(10,35),C1(10,35)
     INTEGER*4 R1(10),K1(35),R2(35),M(3)
     INTEGER*2 NPIL,NFLT,ITER,TAG,I,J,K,KK,K2,K3,K4
     INTEGER*2   D0,D1,D2,S0,S2,P,P1,M0,M1,Q,X0,R
     INTEGER*2 S(10,2),A(10,35)
     INTEGER*2 D(35,2),R3(35),LVAR(2)
     INTEGER*2 XSTAR(45,2),XANDY(45,2),S1(45,2),Y(45,2)
     INTEGER*4 PJ(7),SUMPJ,PJMAX,COST,MCOST
     INTEGER*2 FFEAS,BRANCH,FLAG,FFLAG,NEWMAX
     INTEGER*2 NV,L,LV,PVAR(2),SIP(7,7)
     INTEGER*2 XONE(7),FEAS(7,7),NONE,MSOL(7)
     INTEGER*2 TSOL(7),FM(7),BFEAS(7),FPOINT(35)
     INTEGER*2  F(80),LYR,CPROB(3,300),NF,START,END,QQ

     DATA BIG/3200/
     DATA NEG/-10000/
     DATA LBSTAR/100000/
     DATA LYR/0/
     DATA FFEAS/1/


  OPEN THE FILES

     OPEN(1,FILE='COST.DAT',STATUS='OLD')
     OPEN(2,FILE='OUTPUT.DAT',STATUS='NEW')
     OPEN(3,FILE='FEAS.DAT',STATUS='OLD')
     OPEN(4,FILE='BIP.DAT',STATUS='NEW')

  READ IN THE PROBLEM DATA

9000 FORMAT(1X,A,I5)
     READ(1,1000) NPIL,NFLT
1000 FORMAT(1X,2I5)
     S2=NPIL+1
     D1=NFLT+NPIL
     S0=0
     D0=0
```

citations to applications of this methodology. We will give a general overview here as it relates to the fighter pilot problem.

Lagrangian relaxation is used to provide bounds in a branch and bound algorithm by dualizing some of the constraints. Typically, this procedure is used by constructing a Lagrangian problem that is much easier to solve than the original problem.

In our case we can dualize the node balance constraints, associating Lagrange multipliers v_j with the sink node equations, and multipliers w_i with the supply node equations, giving the "Lagrangian relaxation" problem

$$Z(v,w) = \min \sum_i \sum_j (c_{ij} x_{ij}) + \sum_j v_j (b_j - \sum_i a_{ij} x_{ij}) + \sum_i w_i \ (u_i - \sum_j a_{ij} x_{ij}) \qquad (5.1)$$

subject to

$$\sum_j f_{ikj} x_{ij} \leq 1 \qquad k = 1, \ldots, N, \text{ all } i \qquad (5.2)$$

$$x_{ij} \text{ integer.} \qquad (5.3)$$

We can rewrite the objective function as

```
      DO 10 I=1,NPIL
      READ(1,1010) S(I,1),D(NFLT+I,1)
      D(NFLT+I,2)=D(NFLT+I,1)
      S(I,2)=S(I,1)
 1010 FORMAT(1X,2I5)
   10 CONTINUE
      READ(1,1010) S(S2,1),S(S2,2)

      DO 20 J=1,D1
      READ(1,1020,END=20) (C(I,J),I=1,S2)
 1020 FORMAT(1X,8I5)
      DO 18 K=1,S2
      C1(K,J)=C(K,J)
   18 CONTINUE
   20 CONTINUE

      DO 22 I=1,NFLT
         D(I,1)=1
         D(I,2)=1
   22 CONTINUE

      READ(3,'(1X,I5)') NF
      DO 25 I=1,35,5
         READ(3,'(1X,5I5)') (FPOINT(I+J),J=0,4)
   25 CONTINUE
      DO 30 I=1,NF,5
         READ(3,'(1X,5I5)',END=30) (F(I+J),J=0,4)
   30 CONTINUE

      INITIAL SOLUTION

      D2=0
      K=1
      DO 70 I=1,NPIL
         DO 60 J=I,NFLT,NPIL
            S1(K,1)=I
            S1(K,2)=J
            D2=D2+1
            A(I,J)=D(J,2)
            S(I,2)=S(I,2)-D(J,2)
            D(J,2)=0
            K=K+1
   60    CONTINUE
         S1(K,1)=I
         S1(K,2)=NFLT+I
         D2=D2+1
         A(I,NFLT+I)=S(I,2)
         D(NFLT+I,2)=D(NFLT+I,2)-S(I,2)
         S(I,2)=0
         K=K+1
   70 CONTINUE
```

$$Z(v,w) = \min \sum_i \sum_j (c_{ij} - (w_i + v_j)a_{ij})x_{ij} + \\ (\sum_j v_j b_j + \sum_i w_i u_i). \qquad (5.4)$$

The objective function and constraints (5.2) and (5.3) now separate into M different set covering problems, one for each pilot.

We know that for any solution vector, x^*, which solves the node balance equations is a candidate solution to (5.1) and therefore

$$Z(v,w) \leq \sum cx^* + \sum v(b - \sum ax^*) + \sum w(u - \sum ax^*), \qquad (5.5)$$

where the summations are over the appropriate indices. If x^* is optimal (or even just feasible) to (BIP), then since the equalities in (BIP) must be satisfied, the second and third terms in (5.5) must be zero and, therefore, $Z(v,w) \leq \sum cx^*$. We know that $\sum cx^* = Z$, therefore

$$Z(v,w) \leq Z.$$

A logical goal is to find the values of v and w that maximize Z(v,w), and therefore give us the sharpest lower bound for the value Z of the original problem.

```
      DO 80 J=NFLT+1,D1+DO
         S1(K,1)=S2
         S1(K,2)=J
         D2=D2+1
         A(S2,J)=D(J,2)
         S(S2,2)=S(S2,2)-D(J,2)
         D(J,2)=0
         K=K+1
   80 CONTINUE

      TAG=0


      START TRANSPORTATION ALGORITHM

   90 CONTINUE
      TAG=TAG+1

      DUAL VARIABLE CALCULATION

      ITER=0
 1140 CONTINUE
      ITER=ITER+1
      WRITE(4,1150) ITER
 1150 FORMAT(1X,'ITERATION',I5)
      DO 1160 I=1,D1+DO
         K1(I)=NEG
         R2(I)=10000
 1160 CONTINUE
      DO 1190 I=1,S2+S0
         R1(I)=NEG
 1190 CONTINUE

      R=1
      K=1
      R1(1)=0
      K1(S1(1,2))=C(S1(1,1),S1(1,2))
      GOTO 1240
      R1(S2)=0
      DO 1200 I=D2,D1+2-S2,-1
         IF(S1(I,1).EQ.S2) THEN
            K1(S1(I,2))=C(S1(I,1),S1(I,2))
            K=K+1
            DO 1195 K=1,D2
               IF(S1(K,2).EQ.S1(I,2)) THEN
                  R1(S1(K,1))=C(S1(I,1),S1(I,2))-K1(S1(I,2))
                  R=R+1
               ENDIF
 1195       CONTINUE
         ENDIF
 1200 CONTINUE
```

There are a few methods available for solving for v or w in maximizing Z(v,w). These include subgradient optimization (18), generalized linear programming (for the LP dual problem of maximizing Z(v,w)) (25), and the multiplier adjustment method (10,13). Subgradient optimization has been the dominant procedure used so far, but the new multiplier adjustment method used by Erlenkotter (10) and by Fisher, et al. (13) seems to work much faster in some applications.

The multiplier adjustment method starts with any values of the Lagrange multipliers v and w, which might give a fairly loose lower bound on Z. Then by adjusting each multiplier one by one, we obtain a feasible solution with a much sharper lower bound. This sharper lower bound tends to fathom candidate problems faster than the Ross and Soland method, which we discuss next. See the references for explanations of the procedures discussed so far.

In the next section we discuss a branch and bound method, related to Lagrangian relaxation, developed by Ross and Soland.

5.3 Branch and Bound Algorithm

To solve (BIP), we will use a relaxation algorithm

```
1240  CONTINUE
      I=1
1250  CONTINUE
      I=I+1
      IF(K1(S1(I,2)).NE.NEG) GOTO 1300
      IF(R1(S1(I,1)).EQ.NEG) GOTO 1330
      K1(S1(I,2))=C(S1(I,1),S1(I,2))-R1(S1(I,1))
      K=K+1
1300  CONTINUE
      IF(R1(S1(I,1)).NE.NEG) GOTO 1330
      R1(S1(I,1))=C(S1(I,1),S1(I,2))-K1(S1(I,2))
      R=R+1
1330  CONTINUE
      IF(I.LT.D2) GOTO 1250
      IF(K.LT.D1+DO) GOTO 1240
      IF(R.LT.S2+SO) GOTO 1240

      FIND A VARIABLE TO PIVOT ON

      I=1
      M(1)=0
      DO 1500 R=1,S2+SO
         DO 1490 K=1,D1+DO
            IF(R.NE.S1(I,1)) GOTO 1450
            IF(K.NE.S1(I,2)) GOTO 1450
            IF((A(R,K).EQ.0).AND.
     *         (R2(K).GT.C(R,K)-R1(R)-K1(K))) THEN
               R3(K)=R
            ENDIF
            I=I+1
            GOTO 1490
1450        CONTINUE
            IF(R2(K).GT.C(R,K)-R1(R)-K1(K)) THEN
               R3(K)=R
            ENDIF
            IF(M(1).LT.C(R,K)-R1(R)-K1(K)) GOTO 1490
            M(1)=C(R,K)-R1(R)-K1(K)
            M(2)=R
            M(3)=K
1490     CONTINUE
1500  CONTINUE
      IF(M(1).GE.0) GOTO 2790
      WRITE(4,1502) ITER,M(2),M(3)
1502  FORMAT(1X,'ITER',I5,'PIVOT',2I5)

      FIND A CLOSED PATH FROM R TO K

      Y(1,1)=M(2)
      Y(1,2)=M(3)
      Q=1
      IF(M(2).EQ.S2+SO) GOTO 1960
      MO=Y(Q,1)
      M1=1
```

adapted from Ross and Soland (35). Their algorithm is designed to solve the generalized assignment problem. Our problem structure is such that we can use a slightly modified version of the the algorithm.

5.3.1 Branch and Bound--General

Before discussing the specific aspects of the Ross and Soland method, we review the general principles of branch and bound methods. The general idea is to implicitly enumerate all possible solutions to a problem (such as (BIP)) by cutting the problem in half at each branching step, and then finding the optimal feasible solution for each half.

For instance, we solve a relaxed problem, such as (NET), and find the resulting x^* to be infeasible to (BIP). We select a variable, x_{branch}, to branch on, and split all possible solutions into 2 sets. One set will include all possibilities where $x_{branch} = 1$, and the other set will include all possibilities where $x_{branch} = 0$.

We then solve (NET) again with the stipulation that $x_{branch} = 1$. If the resulting solution is feasible to (BIP) then we know we have the best solution for the $x_{branch} = 1$ branch, and we can focus attention on the solutions where

```
     ROW SEARCH

1610 CONTINUE
     I=0
1620 CONTINUE
     I=I+1
     IF(S1(I,1).GT.M0) GOTO 1670
     IF(S1(I,1).LT.M0) GOTO 1660
     IF(S1(I,2).GE.M1) GOTO 1720
1660 CONTINUE
     IF(I.LT.D2) GOTO 1620
1670 CONTINUE
     IF(Q.NE.1) GOTO 1830
     WRITE(4,8080)
8080 FORMAT(1X,'DEGENERATE MATRIX')
     STOP 'DEGEN'
     CHECK IF ALREADY USED
1720 CONTINUE
     X0=0
     DO 1780 J=1,Q
        IF(S1(I,1).NE.Y(J,1)) GOTO 1780
        IF(S1(I,2).NE.Y(J,2)) GOTO 1780
        X0=1
1780 CONTINUE
     IF(X0.EQ.0) GOTO 1890
     M1=S1(I,2)+1
     IF(M1.LT.D1+D0) GOTO 1660

1930 CONTINUE
     P=Y(Q,2)
     P1=Y(Q,1)+1
     Y(Q,1)=0
     Y(Q,2)=0
     Q=Q-1
     GOTO 2000
1890 CONTINUE
     Q=Q+1
     Y(Q,1)=S1(I,1)
     Y(Q,2)=S1(I,2)
     IF(Q.LE.2) GOTO 1960
     IF(Y(Q,2).EQ.M(3)) GOTO 2340
```

$x_{branch} = 0$.

We then go to (NET) again and solve it when we set $x_{branch} = 0$. Suppose the new solution is not feasible to (BIP). Then we can repeat the branching process on another separation variable. We still include the restriction of $x_{branch} = 0$ along with any new restrictions.

If during this process, any solution to the relaxed problem has an objective value greater than the value of the best feasible solution found so far, we can stop looking for the optimal solution on that the search on a branch. This process of ending branch is called fathoming.

To find the optimum solution to (BIP), we use the branch and bound method until we have fathomed all possible branches. The lowest cost, feasible solution will then be the optimal solution to (BIP).

5.3.2 Ross and Soland Method

This algorithm utilizes a branch and bound framework that first relaxes the time overlap constraints and then solves the network constraints to obtain a candidate solution x^{*}. It then forms small integer problems from the violated time constraints, and solves them to find lower bounds and

```
     COLUMN SEARCH

1960 CONTINUE
     P=Y(Q,2)
     P1=1
2000 CONTINUE
     K=0
2010 CONTINUE
     K=K+1
     IF(S1(K,1).LT.P1) GOTO 2040
2030 CONTINUE
     IF(S1(K,2).EQ.P) GOTO 2120
2040 CONTINUE
     IF(K.LT.D2) GOTO 2010
2050 CONTINUE
     MO=Y(Q,1)
     M1=Y(Q,2)+1
     Y(Q,1)=0
     Y(Q,2)=0
     Q=Q-1
     GOTO 1610

     CHECK FOR UNIQUE PATH SQUARE

2120 CONTINUE
     XO=0
     DO 2180 J=1,Q
        IF(S1(K,1).NE.Y(J,1)) GOTO 2180
        IF(S1(K,2).NE.Y(J,2)) GOTO 2180
        XO=1
2180 CONTINUE
     IF(XO.EQ.0) GOTO 2250
     P1=S1(K,1)+1
     IF(P1.LE.S2+SO) GOTO 2040
     GOTO 2050

     ADD STONE SQUARE TO PATH

2250 CONTINUE
     Q=Q+1
     Y(Q,1)=S1(K,1)
     Y(Q,2)=S1(K,2)
     IF(Q.LE.2) GOTO 2300
     IF(Y(Q,1).EQ.M(2)) GOTO 2340
2300 CONTINUE
     P1=Y(Q,1)+1
     MO=Y(Q,1)
     M1=1
     GOTO 1610
```

separation variables to use in the branching process. We use the separation variables to form candidate problems in which we divide the possibilities in half by adding the constraint that the separation variable must be 1 in our next solution. If the next solution to (NET) (or (BIP)) is feasible, then we try the other half of the possibilities (i.e. solve (NET) when the separation variable is fixed at 0). We first discuss the procedure, then illustrate it with the small example problem formulated in chapter 3.

The relaxed problem is

$$z_R = \min \textstyle\sum_i \sum_j c_{ij} x_{ij} \tag{5.6}$$

subject to

$$\textstyle\sum_i a_{ij} x_{ij} = b_j \qquad \text{all } j \tag{5.7}$$

$$\textstyle\sum_{j'} x_{sj'} = \sum_j b_j - \sum_i l_i \tag{5.8}$$

$$(\text{NET}) \qquad \textstyle\sum_i x_{ij} + x_{sj'} = u_i - l_i \tag{5.9}$$

$$\textstyle\sum_j a_{ij} x_{ij} = u_i \qquad \text{all } i \tag{5.10}$$

$$x_{ij},\ x_{ij'},\ x_{sj'} \text{ integer} \tag{5.11}$$

which is a min-cost flow transportation problem. Later in

```
     FIND THE LEAST FLOW CHANGE

2340 CONTINUE
     XO=A(Y(2,1),Y(2,2))
     DO 2390 K=4,Q,2
        IF(XO.LE.A(Y(K,1),Y(K,2))) GOTO 2390
        XO=A(Y(K,1),Y(K,2))
2390 CONTINUE

     ADD AND SUBTRACT XO ALONG CLOSED PATH

2410 CONTINUE
     P=0
     DO 2450 K=1,Q,2
        A(Y(K,1),Y(K,2))=A(Y(K,1),Y(K,2))+XO
2450 CONTINUE
     DO 2630 K=2,Q,2
        A(Y(K,1),Y(K,2))=A(Y(K,1),Y(K,2))-XO
        IF(A(Y(K,1),Y(K,2)).GT.0) GOTO 2630
        IF(XO.EQ.0) GOTO 2500
        IF((Y(K,2).GT.NFLT).AND.(Y(K,1).LT.S2)) GOTO 2630
2500    CONTINUE

        I=0
        P=P+1
        IF(P.GT.1) GOTO 2630
2530    CONTINUE
        I=I+1
        IF(S1(I,1).NE.Y(K,1)) GOTO 2530
        IF(S1(I,2).NE.Y(K,2)) GOTO 2530
        WRITE(4,8050) Y(K,1),Y(K,2),XO
8050    FORMAT(1X,'PIVOT OUT',2I5,'  FLOW=',I5)
        DO 2590 J=I,D2
           S1(J,1)=S1(J+1,1)
           S1(J,2)=S1(J+1,2)
2590    CONTINUE
        S1(D2,1)=0
        S1(D2,2)=0
        D2=D2-1
2630 CONTINUE

     INSERT A NEW STONE SQUARE

     I=0
2660 CONTINUE
     I=I+1
     IF(Y(1,1).GT.S1(I,1)) GOTO 2660
     IF(Y(1,1).LT.S1(I,1)) GOTO 2700
     IF(Y(1,2).GT.S1(I,2)) GOTO 2660
2700 CONTINUE
```

the chapter we describe methods for solving (NET).

Let x^* denote an optimum flow vector for (NET) and let Z_R denote its optimum objective value. If x^* is feasible for the time constraints, then it is optimal for the original pilot scheduling problem (12).

If the solution x^* to (NET) is infeasible to (BIP), we can then form auxilary problems (subproblems) with the time constraints. We will have one subproblem for each pilot i. The objective of these subproblems is to find the minimum cost reallocation of flights from pilot i to other pilots, so that pilot i's schedule is feasible. By solving these subproblems for all i, we will find a lower bound for Z in (BIP). This lower bound will help fathom the current candidate problem, and help find a separation variable (to use for the next branch).

Let $\bar{c}_{qj}$ be the reduced cost of the pairing of pilot q to flight j in x^*. Let $\bar{c}_{rj}$ be the next larger reduced cost for flight j, and define

$$p_j = \{\bar{c}_{rj} - \bar{c}_{qj}\},$$

then p_j represents the minimum penalty for reassigning flight j with respect to the solution x^*. Also let

$$J_i = \{j : x^*_{ij} = 1\},$$

and

```
      DO 2730 J=D2,1,-1
         S1(J+1,1)=S1(J,1)
         S1(J+1,2)=S1(J,2)
2730  CONTINUE
      S1(I,1)=Y(1,1)
      S1(I,2)=Y(1,2)
      D2=D2+1
      GOTO 1140
2790  CONTINUE
      IF(M(1).GE.0) THEN
         WRITE(4,8120)
8120     FORMAT(1X,'SOLUTION IS OPTIMAL')
      ENDIF


      OPTIMAL SOLUTION, FIND LB

      LB=0
      DO 2800 I=1,D2
         IF(C(S1(I,1),S1(I,2)).LE.0) THEN
            COST=C1(S1(I,1),S1(I,2))
         ELSE
            COST=C(S1(I,1),S1(I,2))
         ENDIF
         LB=LB+(COST*A(S1(I,1),S1(I,2)))
2800  CONTINUE
      DO 2805 I=1,NPIL
         LB=LB+((S(I,1)-D(NFLT+I,1))*C1(I,NFLT+I))
2805  CONTINUE
      WRITE(4,8100) TAG,ITER-1,LB
8100  FORMAT(1X,'TAG',I5,'  ITER',I5,'  LB=',I10)
      IF(LB.GT.BIG+100) GOTO 300
      IF(TAG.GT.40) GOTO 350



      THIS SEGMENT STARTS THE SIP SOLUTION PROCEDURE

      NEWMAX=0
      FFLAG=0
      I=0
 410  CONTINUE
      I=I+1
      NONE=0
      DO 415 K=1,7
         XONE(K)=0
 415  CONTINUE
```

$$y_{ij} = \begin{cases} 1 & \text{if we reassign flight } j \text{ from pilot } i \\ & \text{to pilot } r \\ 0 & \text{otherwise.} \end{cases}$$

Consider the problem

$$z_i = \min \sum_{j \in J_i} p_j \; y_{ij} \tag{5.12}$$

subject to

$$(\mathrm{SIP}_i) \qquad \sum_{j \in J_i} f_{ikj} y_{ij} \geq d_{ik} \qquad \text{all } k \tag{5.13}$$

$$y_{ij} = 0 \text{ or } 1, \tag{5.14}$$

where

$$d_{ik} = \sum_j f_{ikj} \, x^*_{ij} - 1.$$

The value of d_{ik} is the minimum number of flights which must be reassigned to satisfy constraint k. The solution, y^*, this problem represents decisions to as to whether to let pilot i keep flight j (i.e. $y^*_{ij} = 0$), or to reassing flight j to pilot r (i.e. $y^*_{ij} = 1$).

If $y^*_{ij} = 0$, then p_j is large, and we would want to keep this pairing as it is. On the other hand, if $y^*_{ij} = 1$ and p_j is small, we will not be hurt much by reassigning flight j to pilot r.

When we solve (SIP_i) the resulting z_i represents the

```
    J=0
420 CONTINUE
    J=J+1
    IF(S1(J,1).EQ.I) THEN
    XANDY(J,1)=S1(J,1)
    XANDY(J,2)=S1(J,2)
    IF((A(S1(J,1),S1(J,2)).GT.0).AND.
   *   (S1(J,2).LE.NFLT)) THEN
       NONE=NONE+1
       XONE(NONE)=XANDY(J,2)
    ENDIF
    ENDIF
    IF(S1(J,1).LE.I) GOTO 420
    WRITE(4,'(1X,7I5)') (XONE(KK),KK=1,7)

    FILL THE SIP MATRIX

    WRITE(4,9000)'START SIP GEN',I
       DO 440 KK=1,7
          MSOL(KK)=0
          DO 430 K4=1,7
             SIP(KK,K4)=0
430       CONTINUE
440    CONTINUE

    FLAG=0
    KK=0
450 CONTINUE
    KK=KK+1
    START=FPOINT(XONE(KK))
    END=FPOINT(XONE(KK)+1)
    IF(KK.LT.NONE) THEN
    K4=KK
460 CONTINUE
    K4=K4+1
    DO 470 K3=START+1,END-1
       IF(F(K3).EQ.XONE(K4)) THEN
          FFLAG=1
          FLAG=1
          SIP(KK,K4)=1
       ENDIF
470 CONTINUE
    IF(K4.LT.NONE) GOTO 460
    ENDIF
    SIP(KK,KK)=1
    WRITE(4,'(1X,7I5)') (SIP(KK,J),J=1,7)
    IF(KK.LT.NONE) GOTO 450
    WRITE(4,9000)'END SIP MATRIX GEN',I
    WRITE(4,9000)'FLAG=',FLAG
    IF(FLAG.EQ.0) GOTO 725
```

minimum increase in cost by changing x^* to make pilot i's schedule feasible. The overall minimum penalty is $\sum_i z_i^*$, so a lower bound, LB, on (BIP) is

$$LB = Z_R + \sum_i z_i.$$

We can use LB to fathom nodes in the branch and bound procedure (35).

As in Ross and Soland, we can use the solutions y_{ij}^* to suggest a new solution that tends to be feasible. To form the new test solution, we start with the solution x^* from (NET). We then change the x corresponding to $y_{ij}^* = 1$ to zero, and set the corresponding variables variables x_{rj} to one. If this new solution is feasible its objective value is given by LB. The solution is also optimal for the candidate problem we are investigating, since we found the minimum increase in cost when solving the subproblems.

If the new solution is still infeasible, we need to find a separation variable (x_{ij}). A logical choice is one of the variables with $y_{ij}^* = 0$. We choose to branch on the x_{ij} with the maximum p_j for all i. When we branch we will set $x_{ij} = 1$ as the first candidate problem, and $x_{ij} = 0$ as the second.

5.3.3 Algorithm Summary

```
      FIND THE PJ'S

      PJMAX=-5
      WRITE(4,9000)'START SIP SOLUTION',I
      MCOST=-5
      SUMPJ=0
      NV=NONE
      DO 500 J=1,NV
         K2=XONE(J)
         K3=R3(K2)
         PJ(J)=(C(K3,K2)-R1(K3)-K1(K2))-(C(I,K2)-R1(I)-K1(K2))
         SUMPJ=SUMPJ+PJ(J)
500   CONTINUE
      INITIALIZE FEAS
      DO 520 K=1,NV
         DO 510 J=1,NV
            FEAS(K,J)=0
510      CONTINUE
520   CONTINUE
      FILL IN FEAS
      DO 550 J=1,NV
      J=0
525   CONTINUE
      J=J+1
         DO 540 L=1,NV
            IF(SIP(L,J).EQ.1) THEN
               DO 530 K=1,NV
                  IF(SIP(L,K).EQ.1) THEN
                     FEAS(J,K)=1
                  ENDIF
530            CONTINUE
            ENDIF
540      CONTINUE
      IF(J.LT.NV) GOTO 525
550   CONTINUE

      START THE BRANCHING PROCESS

      J=0
555   CONTINUE
      J=J+1
         DO 560 K=1,NV
            TSOL(K)=0
560      CONTINUE
         TSOL(J)=J
         COST=PJ(J)
         LV=J
         FLAG=0
         DO 570 K=1,NV
            BFEAS(K)=FEAS(J,K)
            IF(BFEAS(K).EQ.0) THEN
               FLAG=1
```

To summarize the procedure, figure 5-2 gives the general algorithm, in flow chart form, that we will use to solve the fighter pilot scheduling problem. The following is the written form of the algorithm.

Step 0: Initialize. Read in the data and let $LB^* =$ infinity.

Step 1: Solve (NET)-- using a min-cost network flow algorithm to obtain x^* and Z_R.

Step 2: Test the solution. Test to see if x^* is feasible with respect to the time constraints. If it is feasible or if $Z_R > LB^*$ (the best bound so far), then go to step 6. Otherwise go to step 3.

Step 3: Solve SIP_i for all i. Use an ınteger programming algorithm to find y^* and z_i, and therefore LB for the current candidate problem.

Step 4: Form a new problem--by changing the x variables where $y^*_{ij} = 1$ so that $x_{ij} = 0$ and $x_{rj} = 1$ (r as defined previously). If this new problem is feasible go to step 6, otherwise go to step 5.

Step 5: Select the separation variable. From the

```
          ENDIF
570    CONTINUE
       IF(FLAG.EQ.0) GOTO 600
       BRANCH=0

    FORWARD BRANCH

575    CONTINUE
       K=LV
580    CONTINUE
       K=K+1
       IF(BFEAS(K).EQ.0) THEN
          BRANCH=1
          TSOL(K)=K
          COST=COST+PJ(K)
          LV=K
          FLAG=0
          DO 590 K4=K,NV
             BFEAS(K4)=BFEAS(K4)+FEAS(K,K4)
             IF(BFEAS(K4).EQ.0) THEN
                FLAG=1
             ENDIF
590       CONTINUE
          K=NV
       ENDIF
       IF(K.LT.NV) GOTO 580
       IF((BRANCH.EQ.0).OR.(FLAG.EQ.0)) GOTO 600
       BRANCH=0
       GOTO 575

    BRANCH FATHOMED, CHECK FOR OPTIMUM

600    CONTINUE
       IF(COST.GT.MCOST) THEN
          MCOST=COST
          DO 610 K=1,NV
             IF(TSOL(K).GT.0) THEN
                MSOL(K)=XONE(K)
             ELSE
                MSOL(K)=0
             ENDIF
610       CONTINUE
       ENDIF

    BACKWARD BRANCH

       DO 620 K=LV+1,NV
          BFEAS(K)=BFEAS(K)-FEAS(LV,K)
620    CONTINUE
625    CONTINUE
       IF(TSOL(LV).NE.0) GOTO 630
       LV=LV-1
       IF(LV.LE.0) GOTO 670
       GOTO 625
```

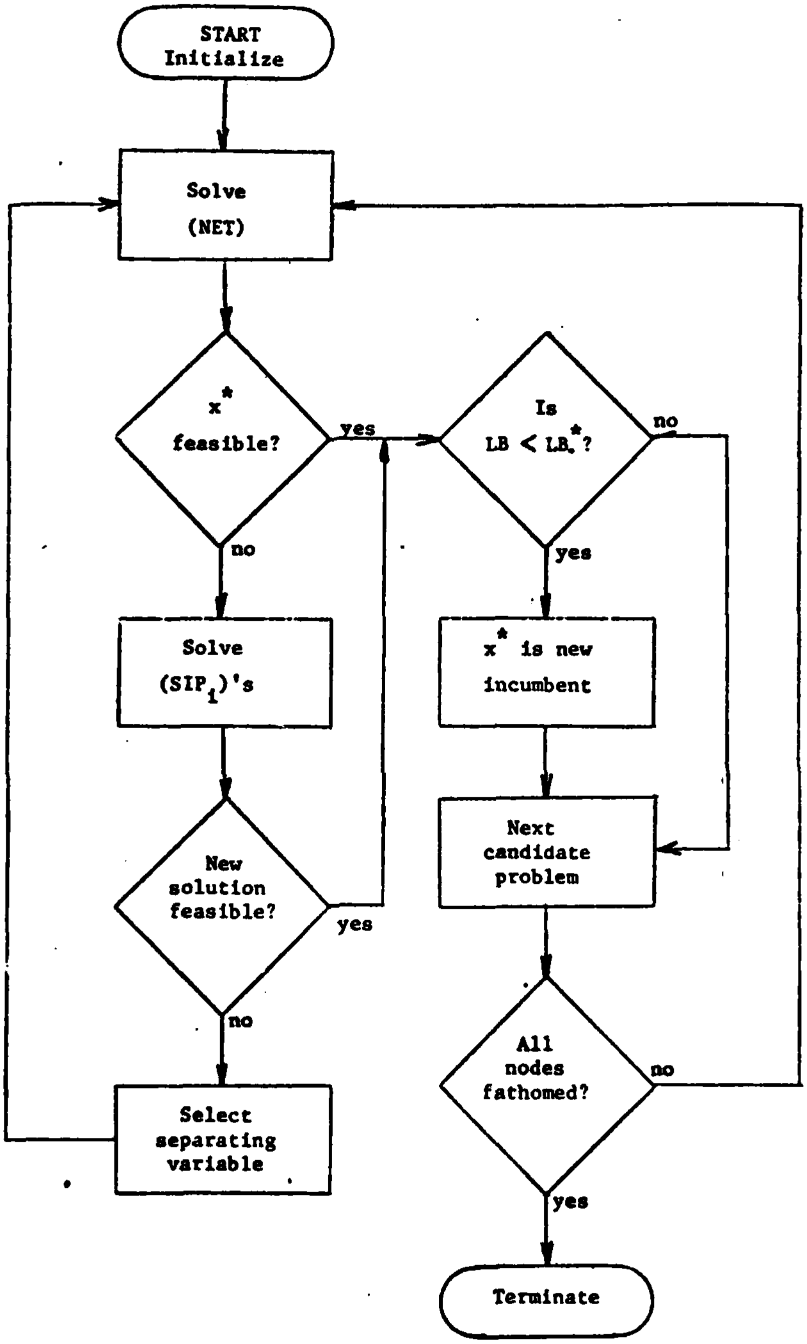


Figure 5-2
Branch and Bound Flow Chart

```
630      CONTINUE
         IF(LV.EQ.J) GOTO 670
         TSOL(LV)=0
         COST=COST-PJ(LV)
         IF(LV.GE.NV) GOTO 600
         BRANCH=0
         GOTO 575
670   CONTINUE
      IF(J.LT.NV) GOTO 555

      WE HAVE THE OPTIMAL SOLUTION FOR THIS I

      WRITE(4,9000)'OPTIMUM FOR SIP',I
      WRITE(4,'(1X,7I5)') (MSOL(KK),KK=1,7)
      J=0
690   CONTINUE
      J=J+1
      IF(MSOL(J).EQ.0) THEN
      FLAG=0
      KK=0
700   CONTINUE
      KK=KK+1
      IF((S1(KK,1).EQ.I).AND.(S1(KK,2).EQ.XONE(J))) THEN
            XANDY(KK,1)=R3(S1(KK,2))
            XANDY(KK,2)=S1(KK,2)
            FLAG=1
      ENDIF
      IF(FLAG.EQ.1) GOTO 705
      IF(KK.LT.D2) GOTO 700
705   CONTINUE
      ENDIF
      IF(J.LE.NV) GOTO 690

      CALCULATE BOUND AND SEPARATION VARIABLE

      LB=LB+SUMPJ-MCOST
      DO 710 K=1,NV
         FLAG=0
         IF(MSOL(K).EQ.0) GOTO 710
         IF(K.EQ.NV) THEN
         DO 706 J=1,NV
            IF(FEAS(J,K).GT.0) THEN
               FLAG=FLAG+1
            ENDIF
706      CONTINUE
         ELSE
         DO 707 J=1,NV
            IF(FEAS(K,J).GT.0) THEN
               FLAG=FLAG+1
            ENDIF
707      CONTINUE
         ENDIF
```

variables where $y^*_{ij} = 0$ select the one with the maximum p_j. Set $x_{ij} = 1$ and go to step 1.

Step 6: Test for optimality. If $LB < LB^*$ then the current solution becomes the new incumbent solution, and let $LB^* = LB$. Go to step 7.

Step 7: Select the next candidate problem. Let the last separation variable (x_{ij}) equal 0, and go to step 1. If there are no more candidate problems, terminate.

This method can be interpreted as Lagrangian relaxation, as the optimal shadow prices, v^* and w^*, from (NET) which determine the reduced costs, c_{ij}, can be viewed as the Lagrange multipliers.

5.3.4 Branch and Bound--Example

We will illustrate the procedure with a simplified example. We consider the example posed in chapter 3, except to help simplify the discussion, we will only use the first four flights (requiring 6 pilots [figure 5-3a]). We assume we have four pilots available, and can model the situation by the network in figure 5-3b. Each pilot must fly at least once, but no more than three times. Figure 5-3c specifies

```
         IF(FLAG.LE.1) GOTO 710
         J=0
 708     CONTINUE
         J=J+1
         IF((CPROB(1,J).EQ.I).AND.
     *      (CPROB(2,J).EQ.XONE(K))) GOTO 710
         IF(J.LT.LYR) GOTO 708
         IF(PJMAX.LT.PJ(K)) THEN
            PJMAX=PJ(K)
            PVAR(1)=I
            PVAR(2)=XONE(K)
            NEWMAX=1
         ENDIF
 710  CONTINUE
 725  CONTINUE
      WRITE(4,8200) I
8200  FORMAT(1X,'PILOT',I3,'FATHOMED')
      IF(I.LT.NPIL) GOTO 410
      IF(FFLAG.EQ.0) GOTO 280
      IF(NEWMAX.EQ.0) GOTO 300
      IF((LVAR(1).EQ.PVAR(1)).AND.
     *   (LVAR(2).EQ.PVAR(2))) GOTO 300
      LVAR(1)=PVAR(1)
      LVAR(1)=PVAR(2)

      TEST TO SEE IF XANDY IS FEASIBLE

      FFLAG=0
      I=0
      K=0
 210  CONTINUE
      I=I+1
      DO 215 R=1,7
         FM(R)=0
 215  CONTINUE
      DO 220 J=1,D2
         IF((XANDY(J,1).EQ.I).AND.(XANDY(J,2).LE.NFLT)) THEN
         K=K+1
         FM(K)=XANDY(J,2)
         ENDIF
 220  CONTINUE

      KK=0
 230  CONTINUE
      KK=KK+1
      START=FPOINT(FM(KK))
      END=FPOINT(FM(KK)+1)
      IF(KK.LT.K) THEN
      K4=KK
 240  CONTINUE
      K4=K4+1
      K3=START
 250 CONTINUE
     K3=K3+1
     IF(F(K3).EQ.FM(K4)) THEN
        FFLAG=1
        KK=K
        K3=END-1
        K4=K
     ENDIF
     IF(K3.LT.END-1) GOTO 250
     IF(K4.LT.K) GOTO 240
     ENDIF
     IF(KK.LT.K) GOTO 230
     IF((I.LT.NPIL).AND.(FFLAG.EQ.0)) GOTO 210
     WRITE(4,9000)'FFLAG XANDY=',FFLAG
     IF(FFLAG.EQ.1) GOTO 320

     CHECK TO SEE IF XANDY IS OPTIMAL TO BIP

     IF(LB.LT.LBSTAR) THEN
        LBSTAR=LB
        DO 270 J=1,D2
           XSTAR(J,1)=XANDY(J,1)
           XSTAR(J,2)=XANDY(J,2)
 270    CONTINUE
     ENDIF
     IF(FFEAS.EQ.1) GOTO 350
     GOTO 300

     CHECK IF S1 IS OPTIMAL

 280 CONTINUE
     IF(LB.LT.LBSTAR) THEN
        LBSTAR=LB
        DO 290 J=1,D2
           XSTAR(J,1)=S1(J,1)
           XSTAR(J,2)=S1(J,2)
 290    CONTINUE
     ENDIF
     IF(FFEAS.EQ.1) GOTO 350

     OVERALL BRANCH AND BOUND CONTROL

     ELIMINATE VARIABLES

 300 CONTINUE
     WRITE(4,8030) TAG
8030 FORMAT(1X,'TAG',I5,'  ELIMINATE VARS')
 310 CONTINUE
```

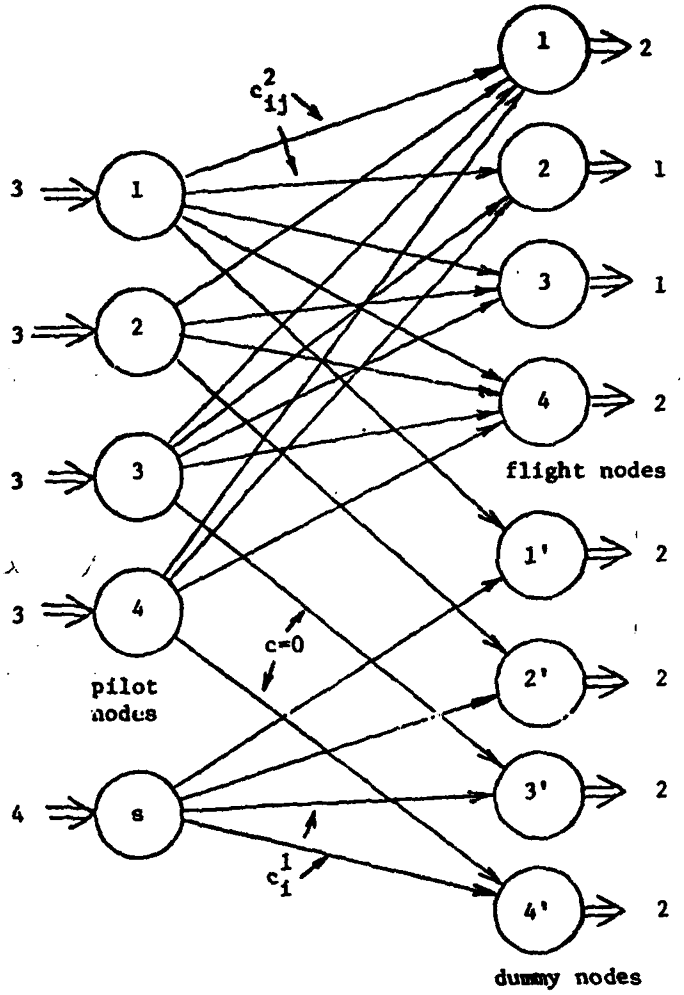


Figure 5-3a

Network Representation of the Sample Problem

```
      IF(LYR.EQ.0) GOTO 350
      FLAG=0
      J=CPROB(1,LYR)
      K=CPROB(2,LYR)
      IF(CPROB(3,LYR).EQ.0) THEN
         C(J,K)=C1(J,K)
         CPROB(1,LYR)=0
         CPROB(2,LYR)=0
         LYR=LYR-1
         FLAG=1
      ELSE
         C(J,K)=BIG
         CPROB(3,LYR)=0
      ENDIF
      IF(FLAG.EQ.1) GOTO 310
      GOTO 90

      ADD NEW VARIABLES

  320 CONTINUE
      WRITE(4,8040) TAG,PVAR(1),PVAR(2)
 8040 FORMAT(1X,'TAG',I5,'  ADD VAR',2I5)
      LYR=LYR+1
      CPROB(1,LYR)=PVAR(1)
      CPROB(2,LYR)=PVAR(2)
      CPROB(3,LYR)=1
      C(PVAR(1),PVAR(2))=-3200
      ADD ZERO VARIABLES
      IF(PVAR(2).LT.11) THEN
         QQ=PVAR(2)-1
      ELSE
         QQ=10
      ENDIF
      DO 327 K=PVAR(2)-QQ,PVAR(2)-1
         DO 323 J=FPOINT(K)+1,FPOINT(K+1)-1
            IF(F(J).EQ.PVAR(2)) THEN
               LYR=LYR+1
               CPROB(1,LYR)=PVAR(1)
               CPROB(2,LYR)=K
               CPROB(3,LYR)=0
               C(PVAR(1),K)=BIG
            ENDIF
  323    CONTINUE
  327 CONTINUE
```

Day 1	Flight 1	Flight 2	Flight 3
Brief time	0515	0930	1400
Takeoff time	0715	1130	1600
Type flight	Air Combat	DART	Night Inter
	2 pilots required	1 pilot required	1 pilot required
Land time	0830	1245	1715
End debrief time	1015	1430	1900

Day 2	Flight 4
Brief time	0500
Takeoff time	0700
Type flight	Air Refuel
	2 pilots required
Land time	0815
End debrief time	1000

Figure 5-3b
Example Problem Schedule

```
      START=FPOINT(PVAR(2))
      END=FPOINT(PVAR(2)+1)
      DO 330 J=START+1,END-1
         I=PVAR(1)
            LYR=LYR+1
            CPROB(1,LYR)=I
            CPROB(2,LYR)=F(J)
            CPROB(3,LYR)=0
            C(I,F(J))=BIG
  330 CONTINUE
      GOTO 90

      OPTIMAL SOLUTION IS REACHED

  350 CONTINUE
      DO 360 I=1,D2
         IF((A(XSTAR(I,1),XSTAR(I,2)).GT.0).AND.
     *      (XSTAR(I,2).LE.NFLT)) THEN
         WRITE(2,'(1X,3I10)') XSTAR(I,1),XSTAR(I,2),
     *                        A(XSTAR(I,1),XSTAR(I,2))
         ENDIF
  360 CONTINUE
      LB=0
      DO 370 I=1,D2
         LB=LB+(C1(XSTAR(I,1),XSTAR(I,2))*A(XSTAR(I,1),
     *            XSTAR(I,2)))
  370 CONTINUE
      WRITE(2,8000) LBSTAR
 8000 FORMAT(1X,I10,' = LBSTAR')
      WRITE(2,8010) TAG
 8010 FORMAT(1X,I10,' = NO. OF ITERATIONS')
      STOP
      END
```

C.3 Program to Format Schedule

```
      PROGRAM OUTPUT

      INTEGER*2 FIL(300,2),FLTN(150,5),NUMF,X(150)
      INTEGER*2 FLAG,NPIL,NFLT,PER,DAY,FLT,N
      CHARACTER*4 TYPE(150),PNAME(35)

      OPEN(1,FILE='OUTPUT.DAT',STATUS='OLD')
      OPEN(2,FILE='PILOT.DAT',STATUS='OLD')
      OPEN(3,FILE='SCHED.DAT',STATUS='OLD')
      OPEN(4,FILE='BYNAME.DAT',STATUS='NEW')

      READ(2,1000) NPIL
 1000 FORMAT(//I5)
      READ(3,'(//I5)') NFLT

      DO 10 I=1,NPIL
      READ(2,1020) PNAME(I)
 1020 FORMAT(A10)
   10 CONTINUE

      DO 20 J=1,NFLT
      READ(3,1030) TYPE(J),(FLTN(J,K),K=1,5)
 1030 FORMAT(3X,A5,5X,5I5)
   20 CONTINUE

      DO 30 K=1,NFLT
      READ(1,1040) FIL(K,1),FIL(K,2)
 1040 FORMAT(1X,2I10)
   30 CONTINUE

      FLT=0
   35 CONTINUE
      FLT=FLT+1
      K=0
      FLAG=0
   37 CONTINUE
      K=K+1
      IF(FIL(K,2).EQ.FLT) THEN
         X(FLT)=FIL(K,1)
         FLAG=1
      ENDIF
      IF(FLAG.EQ.1) GOTO 35
      IF(K.LT.NFLT) GOTO 37
      IF(FLT.LT.NFLT) GOTO 35
```

the cost (c_{ij}) and time overlap (f_{kj}) matricies, that we developed in chapter 3. An "X" in the cost matrix means that the pilot cannot fly that flight (due to other obligations).

Step 0: Initialize. LB^* = infinity.

Step 1: The optimal solution is the set of pairings shown circled in figure 5-4a. $Z_R = 9$.

Step 2: Pilot 4's schedule is infeasible since he is to fly both flights 1 and 2, so we go to step 3.

Step 3: We find the p_j's by looking at figure 5-4a and noting that to reassign flight 1 from pilot 4 to pilot 1 would cost nothing, and to reassign flight 2 to pilot 3 would cost 2 units. We then solve SIP_4 and find $y^*_{41}= 1$, and $y^*_{42}= 0$ (figure 5-4b). LB = 9.

Step 4: The new solution, after reassigning flight 1, is still not feasible.

Step 5: We choose x_{42} as the separation variable, so we set $x_{42} = 1$, $x_{41} = 0$, (we know x_{41} cannot equal 1 in a feasible solution). Go to step 1.

Mathematical Programming, Addison-Wesley Publishing Co., Reading, MA, 1977.

8. Cohon, J.L., Multiobjective Programming and Planning, Academic Press, New York, NY, 1978.

9. Conway, R.W., Maxwell, W.L., and Miller, L.W., Theory of Scheduling, Addison-Wesley Publishing Co., Reading, MA, 1967.

10. Erlenkotter, D., "A Dual Based Procedure for Uncapacitated Facility Location", Operations Research, Vol. 26, No. 6, pp. 992-1009, (1978).

11. Etcheberry, J., "The Set-Covering Problem: A New Implicit Enumeration Algorithm", Operations Research, Vol. 25, pp. 760-772, (1977).

12. Fisher, M.L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", Management Science, Vol. 27, No. 1, pp. 1-18, (1981).

13. ____________, Jaikumar, R., and Van Wassenhove, L.N., "A Multiplier Adjustment Method for the Generalized Assignment Problem", Department of Decision Sciences Working Paper, University of Pennsylvania, (1981).

14. Ford, L.R. and Fulkerson, D.R., Flows in Networks, Princeton University Press, Princeton, NJ, 1962.

Flights

	c_{ij}	1	2	3	4
	1	2	4	①	3
Pilots	2	3	☒	2	②
	3	①	3	2	2
	4	②	①	☒	②

$Z_R = 9$

Pilot 4 is infeasible

Soution to (NET) - no restrictions

$$z_4 = \min \quad 0y_{41} + 2y_{42}$$
$$\text{subject to} \quad y_{41} + y_{42} \geq 1$$

$z_4 = 0$, $y^*_{41} = 1$, $y^*_{42} = 0$

$LB = Z_R + z_4 = 9$

Figure 5-4a

Example Problem--First Solution

15. Garfinkel, R.S. and Nemhauser, G.L., "The Set Partitioning Problem: Set Covering with Equality Constraints," Operations Research, Vol. 17, No. 5, pp. 848-856, (1969).

16. ______________ and ______________, Integer Programming, John Wiley and Sons, New York, NY, 1972.

17. Golden, B.L. and Magnanti, T.L., Network Optimization, John Wiley and Sons (to appear).

18. Held, M., Wolfe, P., and Crowder, H.P., "Validation of Subgradient Optimization", Math Programming, North-Holland Publishing Co., Vol. 6, pp. 62-68, (1974).

19. Hogan, W.W., Marsten, R.E., and Blankenship, J.W., "The Boxstep Method for Large Scale Optimization", Operations Research, Vol. 23, p. 3, (1975).

20. Ignizio, J.P., Goal Programming and Extensions, D.C. Heath and Co., Lexington, MA, 1976.

21. Kennington, J.L. and Helgason, R.V., Algorithms for Network Programming, John Wiley and Sons, New York, NY, 1980.

22. Laffin, J., Fight for the Falklands, St. Martins Press, New York, NY, 1982.

23. Levin, R.I., Kirkpatrick, C.A. and Rubin, D.S.,

Step 1: The solution to the candidate problem with $x_{42} = 1$ is in figure 5-5a. $Z_R = 9$.

Step 2: Pilot 1's schedule is now infeasible because he is scheduled for flights 1 and 3.

Step 3: We solve SIP_1 and find $y^*_{11} = 1$, $y^*_{13} = 0$, and $LB = 10$.

Step 4: Reassigning flight 1 to pilot 2 yields a feasible solution (figure 5-5b), so this candidate problem is fathomed, and we go to step 6.

Step 6: 10 is less than infinity, so $LB^* = 10$, and the candidate problem with $x_{42} = 1$ is the current incumbent

Quantitative Approaches to Management, McGraw-Hill, New York, NY, 1982.

24. Loomba, N.P. and Turban, E., Applied Programming for Management, Holt, Rinehart, and Winston, New York, NY, 1974.

25. Magnanti, T.L., "Optimization for Sparse Systems", Sparse Matrix Computations, D.Rose and J. Bunch, eds., Academic Press, New York, NY, 1976.

26. Marsten, R.E., "An Algorithm for Large Set Partitioning Problems", Management Science, Vol. 20, No. 5, pp. 774-787, (1974).

27. Miller, H.E., "Personnel Scheduling in Public Systems: A Survey", Socio-Economic Planning Sciences, Vol. 10, No. 6, pp. 241-249, (1976).

28. ________, Pierskalla, W.P. and Rath, G.J., "Nurse Scheduling Using Mathematical Programming", Operations Research, Vol. 24, No. 5, pp. 857-870, (1976).

29. Muth, J.F. and Thompson, G.L., eds., Industrial Scheduling, Prentice-Hall, New York, NY, 1963.

30. Moreland, J.A., "Scheduling of Airline Flight Crews", MS thesis, Department of Aeronautics and Astronautics, Sept 1966.

31. Nanney, T.R., Computing: A Problem Solving Approach with Fortran 77, Prentice-Hall, Englewood Cliffs, NJ, 1981.

32. Nickoletti, B., "Automatic Crew Rostering", Transportation Science, Vol. 9, No. 1, pp. 33-42, (1975).

33. Pierce, J.F., "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems", Management Science, Vol. 15, pp. 191-209, (1968).

34. Poole, L., ed., Practical Basic Programs, Osborne/McGraw-Hill, Berkeley, CA, 1980.

35. Ross, G.T. and Soland, R.M., "A Branch and Bound Algorithm for the Generalized Assignment Problem", Math Programming, Vol. 8, pp. 91-103, (1975).

36. Rubin, J., "A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling", Transportation Science, Vol. 7, No. 1, pp. 34-48, (1973).

37. Shapiro, J.F., "Dynamic Programming Algorithms for the Integer Programming Problem - I: The Integer Programming Problem Viewed as a Knapsack Type Problem", Operations Research, Vol. 16, NO. 1, pp. 103-121, (1968).

38. ____________, Mathematical Programming, Structures and Algorithms , John Wiley and Sons, New York, NY, 1979.

Flights

Pilots

$$\begin{array}{c|cccc}
c_{ij} & 1 & 2 & 3 & 4 \\
\hline
1 & \cancel{\textcircled{2}} & 4 & \textcircled{1} & 3 \\
2 & \textcircled{3} & \times & 2 & \textcircled{2} \\
3 & \textcircled{1} & 3 & 2 & 2 \\
4 & \cancel{2} & \textcircled{1} & \times & \textcircled{2}
\end{array}$$

LB = 10
Solution is
feasible
$LB^* = 10$

Figure 5-5b
Solution After Second Reassignment

39. _____________, "A Survey of Lagrangian Techniques for Discrete Optimization", Annals of Discrete Mathematics, Vol. 5, pp. 113-138, (1979).

40. Shepardson, W.B., "A Lagrangian Relaxation Algorithm for the Two-Duty Period Scheduling Problem", Ph.D Dissertation at the Massachusetts Institute of Technology, Alfred P. Sloan School of Management, June 1978.

41. Tactical Air Command Manual 51-50, Vol. I, Department of the Air Force, Headquarters Tactical Air Command, Langley AFB, VA, 26 Oct. 1981.

Flights

Pilots c_{ij}	1	2	3	4
1	2	4	①	3
2	3	☒	2	②
3	①	③	2	2
4	②	☒	☒	②

$Z_R = 11$

$Z_R > LB^*$

so the problem
is fathomed

Solution to (NET) with $x_{42} = 0$

Figure 5-6

Example Problem—Third Solution

FILM

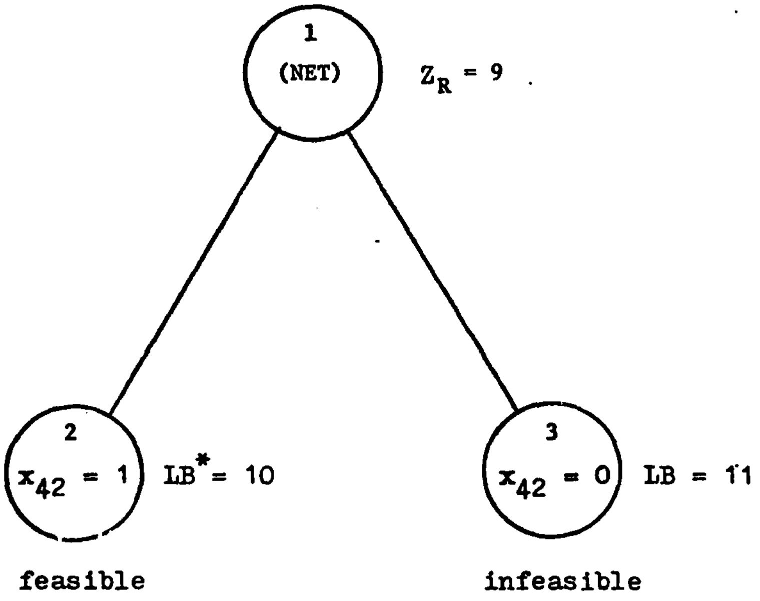


Figure 5-7
Branch and Bound Summary

Step 7: There are no more candidate problems, so terminate. The optimal solution is $x_{21} = 1$, $x_{31} = 1$, $x_{42} = 1$, $x_{13} = 1$, $x_{24} = 1$, and $x_{44} = 1$, with $Z = 10$.

This example showed how we may be able to find a feasible solution by reassigning flights when $y^* = 1$, and that we can fathom candidate problems by use of the best lower bound. Figure 5-7 gives a picture of how we used the branch and bound process.

5.4 Network Problem

To find candidate solutions for x to use in the (SIP_i)'s, we must solve an assignment type min-cost network flow problem. We have three possible solution methods: the primal simplex (7), the primal-dual (5,6), and the out-of-kilter (14). See the references for explanations of the primal-dual and out-of-kilter methods.

The primal simplex method has been modified for use with min-cost network and transportation problems (17,23). The program we will use is a specialized version of the simplex method called the modified distribution method, which is used for transportation problems. Our code was adapted from Levin, Kirkpatrick, and Rubin (23), and Poole (34). The algorithm finds augmenting paths at each pivot, and then pivots the new variable into the basis. We can use the "big M" method for our cost structures (i.e. infeasible pairings will have very large costs) so that we do not need to start with a feasible solution. Any solution that satisfies the supply and demand constraints (even over infeasible arcs) will serve as a starting solution. We can use the big M property to advantage during our branching process. When we set $x_{ij} = 0$ we change c_{ij} to big M and it is pivoted out of the basis. Similarly, if we wish x_{ij} to be 1, we let $c_{ij} = -M$ and x_{ij} is pivoted into the basis. We can then start the intermediate solution process from an almost feasible (and almost optimal) solution. The time required for such a solution procedure is shorter than if we solved the new problem from scratch at each iteration.

The algorithm is explained in detail in Levin, et al (23), and in many Operations Research texts. Poole (34) gives a BASIC code for the algorithm.

5.5 Time Constraint Subproblems

The final section of this chapter describes the methodology we can use to solve the subproblem (SIP_i) formulated earlier. There are two methods we will consider for possible use. The first is to convert (SIP_i) into a knapsack problem and then, using knapsack algorithms, find a solution, or second, because the problem is small, we can enumerate the solutions and select the best one.

5.5.1 Knapsack Solution Method

Shepardson (40) and Garfinkel and Nemhauser (16) show two different methods for converting multiple constraints to a single constraint.

Shepardson uses a prime number technique that will take a set of constraints such as the time overlap constraints in (BIP), and combine them into a single constraint. For example, the constraints

$$\textstyle\sum_{j=1}^{N} f_{ikj} y_{ij} + s_k = 1 \quad \text{for } k = 1, 2, \ldots, K, \qquad (5.15)$$

forms the single constraint

$$\textstyle\sum_{j=1}^{N} \sum_{k=1}^{N} (f_{kj} \ln P_k) y_{ij} + \sum_{j=N+1}^{2N} (\ln P_k) s_k = \sum_{j=1}^{N} (\ln P_k), \qquad (5.16)$$

where P_k equals the k th prime number. The main shortcoming of this method is that the $\ln P_k$ are normally irrational numbers which must be appropriately approximated to find a solution. As a result, the numbers in the problem may become very large.

Garfinkel and Nemhauser describe a method which combines constraints in pairs until all are combined into one constraint. Suppose we want to combine the constraints

$$\sum_{j=1}^{N} f_{1j}y_{ij} + s_1 = 1, \tag{5.17}$$

and $$\sum_{j=1}^{N} f_{2j}y_{ij} + s_2 = 1 \tag{5.18}$$

into one.

We first find a multiplication factor, α, for one constraint (say the first). We then multiply the other constraint by α, and then add the two constraints together. In our problem we can always weight the constraints by $\alpha = \sum f_{ikj} + 1$ (refer to Garfinkel and Nemhauser). The new constraint is given by

$$\sum_{j=1}^{N} (f_{1j}+\alpha_1 f_{2j})y_{ij} + s_1 + \alpha_1 s_2 = 1 + \alpha_1. \tag{5.19}$$

We can then combine the new equation with another equation, and repeat the process until only one constraint remains. If we had a large number of constraints, this method could produce some large numbers, but with our problem size the derived coefficients should not be excessively large.

Once we transform the set covering constraints to knapsack constraints we can solve the problem by efficient dynamic programming algorithms. Garfinkel and Nemhauser (16) give an algorithm that is appropriate for solving this problem.

5.5.2 Enumeration

Because of the small size of (SIP_1), enumeration might be almost as fast as using a knapsack algorithm. Even though the problem might have a large number of feasible solutions, on the average we would expect the problems to be very small, and solution times very small. We also eliminate the time required to transform the problem. Therefore we will use the enumeration technique when implementing the solution procedure.

CHAPTER 6

CONCLUSION

6.1 Background

Our goal in this thesis has been to develop a model that would solve the fighter pilot problem on a micro-computer. We did not set out to develop a computer code that is in any sense best, or even efficient. Rather, we wished to establish the computational viability of using micro-computers and modern integer programming methods to solve scheduling applications such as the squadron pilot problem. Therefore, most of our observations are geared toward the problem structure, implementation issues, and a general evaluation of the method.

In order to ensure that the program would run on a micro- computer, we developed and tested our code on the IBM personal computer (IBM PC). Our particular computer was equipped with a FORTRAN 77 compiler that we decided to use for this project. The IBM PC contained 128K of internal memory and 2-320K, 5 1/4" disk drives.

To test the program we obtained old schedules from the 27th Tactical Fighter Squadron to use as the data. We then used a subset of the data for the development and initial stages of testing. We never progressed far enough to try full size problems.

6.2 Methodology

Our approach to the problem was to solve it in 3 phases: a matrix generation phase, an optimization phase, and an output phase.

The matrix generation phase takes the raw data from user data files and converts the data into a cost matrix and a feasibility matrix (as we did in the example in Chapter 3). We puc chese two matrices inco files, as inputs co the optimization phase.

We had five raw data files:

1. Pilot data -- this includes the pilot's name and qualifications data,

2. Pilot accomplishment -- this file contains the number of each type of flight a pilot has flown,

3. Pilot availability -- this file contained information concerning when a pilot was not to be available for flying duty (day and times),

4. Requirement data -- this file stores the TACM 51-50 requirements,

5. Schedule -- this file holds the schedule we wish to fill. It includes times, type of flight, and the qualifications required to fly it.

The optimization phase solved the problem using a branch and bound algorithm as we have discussed in Chapter 5. We originally tried to use a general network simplex algorithm (the code was called NETFLO [21]) to solve the relaxed network problem. The code proved to be too large for the IBM PC when imbedded in the branch and bound code. We then decided to use a code designed to solve the classical Hitchcock transportation problem (34).

The code to solve the subproblems is an enumeration method. We first develop a matrix that indicates which pairings are infeasible, so we do not have to consider all possible solutions to the problem.

The branch and bound code directs the program flow and keeps track of the current candidate problem. It puts bounds on the variables by changing costs depending on whether we want the variable at 1, 0, or free (e.g., cost equals "M" if the variable is restricted to zero or equals "-M" if the variable is restricted to 1).

We use a depth first search to find a feasible solution quickly. If we find a feasible solution early in the enumeration procedure, we can reduce the number of problems to be considered. We also include the option of stopping at the first feasible solution, which might be useful for problems that are too large to solve to optimality or for problems where we obtain "good" or near optimal solutions before terminating the complete branch and bound eumeration.

At each branch we use the feasibility matrix (as in the example problem) to exclude all variables that conflict with the separation variable. This hopefully helps lead to a feasible solution. If our transportation algorithm then yields a solution that includes infeasible arcs, we know there are no feasible solutions along that branch, so we can fathom the branch.

Once it has discovered the solution to the problem, the program writes it into a file for the output generation phase.

The output generation phase contains a short program to sort the solution and display it in a form useful to the user.

Appendix C contains the computer code of the 3 programs.

6.3 Results

Our first concern was that the cost structure would lead to unstable solutions. Many of the flight categories have requirements for only 2 to 4 flights (e.g., DART and INST) and in our data many pilots had not accomplished any, meaning that many of the costs were essentially zero. We were concerned that this degeneracy would have a serious effect on our ability to obtain a solution.

We found, in the transportation algorithm, that 70 per cent of the pivots were degenerate, in that they involved no transfer of flow. They only moved variables in and out of the basis. The algorithm did, however, find optimal solutions each time it was used.

This means that the subproblems consumed the major share of the solution time. Reducing the solution time would require an efficient algorithm for the subproblems (such as a good 0-1 knapsack algorithm).

Another finding was that the number of pilots unavailable to fly due to other commitments had a significant impact on the ability to find a feasible solution (to BIP). Problems with relatively few instances of unavailable pilots were solved much faster than problems where pilots had numerous other duties.

The internal memory of the IBM PC is capable of handling our program and data. The storage required for an 8 by 25 problem is only 6.5K. The execution code requires 56K of storage.

6.4 Conclusion

The methods we have discussed do solve the fighter pilot scheduling problem. There is, however, room for improvement. The computer code could be improved to accelerate computations. There may be better algorithms (such as the more complicated multiplier adjustment method) to solve the problem. In the future, we hope to see if any of these methods can be successfully implemented on a micro-computer.

Let us analyze our program with respect to the goals we set for ourselves in Chapter 2. The first goal is to ensure that TACM 51-50 flight requirements are met. We accomplish this through our objective function. Our costs are such that, those pilots who are behind relative to other pilots will be scheduled more often. Although this approach does not ensure all flight requirements will be met, it does tend to keep anyone from lagging behind. Moreover, it gives the schedulers the flexibility to change scheduling priorities for the pilots by changing the cost structure.

The second goal is to ensure that each pilot's minimum and maximum number of flights per week are observed. Our transportation algorithm, by virtue of our lower and upper bound transformations ensures that we comply with this restriction.

The third goal is to ensure no pilot flies without proper rest, flies with too long a duty day, or is scheduled when not available to fly. Our development of the overlap constraints and the feasibility matrix ensure that no one is scheduled during those times.

The fourth objective is to solve the problem in less time than the present system. The present system takes about two man-days of work to find a "good" schedule. Once proficient wich the data structures, schedulers could solve the problem in less than 1 hour, including inputting data into the data files and running the program. Clearly, using this program would provide time savings for the schedulers and free them for other tasks.

The fifth goal is to run the program on a micro-computer. We have successfully accomplished this, however, we have not tried full-scale problems yet. The storage requirements for our sample problems were well within the capabilities of the IBM PC, and we postulate that we could, in fact, solve problems of 30 pilots and 120 flights on this computer.

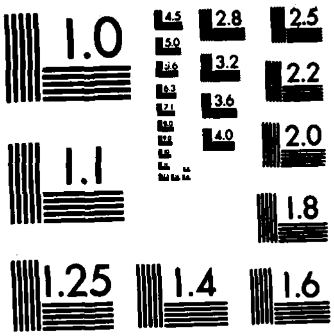
We did well on the five goals we stated, but we also mentioned that we would like to have auxiliary programs that are useful in daily decision making. We were not successful on this point as time did not permit us to concentrate on that aspect of the model. In addition to efforts in bettering the optimization code, we would like to see someone develop a user friendly interface with the program, so that non-technical people could effectively run the optimization.

AD-A126 947 PILOT SCHEDULING IN A FIGHTER SQUADRON(U) AIR FORCE 2/2
INST OF TECH WRIGHT-PATTERSON AFB OH W H ROEGE FEB 83
AFIT/CI/NR-83-2T

UNCLASSIFIED F/G 5/9 NL

END

FILMED



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

We are convinced that the use of Operations Research and Computer Science planning tools, such as those discussed in this thesis, are of great benefit to the Air Force. Specifically, we believe that these tools can be used at the Squadron and Wing levels, not only for pilot scheduling, but for many of a number of similar scheduling and allocation problems.

APPENDIX A

FLIGHT TYPES

Air Combat Training (ACTT).

These are missions where similar types of aircraft practice "dogfight" maneuvers against each other. Weapons launches and weapons parameters are simulated and evaluated with gun camera film (42 of these flights are required every 6 months).

Dissimilar Air Combat Training (DACT).

These missions are the same as ACTT, except they are flown against other types of aircraft (DACT flights are included in the ACTT requirements).

Airborn Gunnery Practice (DART).

This mission involves firing the 20MM cannon at a metal target (Dart) which is towed 1500 feet behind another aircraft (1 or 2 of these missions are required depending on the pilot's experience level).

Intercept Training (DINT).

Intercept training involves using electronic means (e.g. RADAR) to find and simulate firing on a target. Maneuvers are much more restricted than in ACTT or DACT due to the limitations of the equipment (5 or 6 of these missions are required depending on the pilot's experience level).

Night Intercept Training (NINT).

Night intercepts are the same as day intercepts, except they must be performed at night (4 are required per 6 month period).

Air to Air Refueling (AARD).

A specially modified Boeing 707 or DC-10 carries fuel and the fighters practice intercepting the "tanker" and taking on gas through an 18 foot long "boom" on the tail end of the tanker (2 required).

Night Air to Air Refueling (NAAR).

Night air to air refueling is the same as day refueling except that it must be accomplished at night (1 required).

Instrument Proficiency Flights (INST).

These flights are dedicated to practicing instrument approaches and other instrument procedures. The are only required for non-experienced pilots (2 every 6 months).

APPENDIX B

ADDITIONAL DUTIES

Supervisor of Flying (SOF).

Only Lt Colonels, Majors, and very senior Captians who are experienced pilots may serve as SOF. The SOF sits in the control tower, and is responsible for the entire flying operations of the Wing. He has the authority to cancel flights due to weather or other circumstances. He also is there to assist any aircraft in time of an emergency, since he can call on other aircraft, fire trucks, and other resources for help.

Runway Supervisory Officer (RSO).

All MR pilots are qualified to serve as RSO. SOF's are qualified, but do not serve as RSO. The RSO serves in a special building near the end of the runway. He ensures the landing patterns are safe and that everyone lands with their landing gear down. He can also assist in emergencies by looking over the emergency aircraft for obvious exterior problems when it flies by.

Range Training Officer (RTO).

RTO's must be MR and have some experience. Approximately half the pilots are qualified to be RTO's. The RTO monitors flights which fly on a range where ground stations receive flight information from aircraft and feed the information into a computer. The computer then displays the flight on a video screen. The RTO can see a "God's eye" view of the live action and warn pilots of any dangers. The information is stored, and can be replayed in the flight debrief. The RTO monitors the live flight for safety, simulates missle launches in the computer, and relates the missle results to the fliers.

APPENDIX C

COMPUTER CODES

These codes were written in FORTRAN 77 for the IBM personal computer.

The first program converts the raw data from the data files into the cost and feasibility matrices.

The second program is the optimization program that takes the cost and feasibility data and outputs the optimal schedule.

The third program is a short program to format the output as an easy to read document.

C.1 Program to Organize Raw Data into Problem Data

```
     PROGRAM FILGEN
  THIS PROGRAM TAKES THE RAW DATA FILES
    AND PROCESSES THEM TO DATA THE PILOT
    OPTIMIZATION PROGRAM CAN USE.
     INTEGER*2 FEAS(1200),P(30,2),FPOINT(150),C(30,150),
    *ACC(30,9),AVL(30,10,4),REQ(3,9),S(150,4),SCH(150,3),
    *NE(30),ENDDAY(5),NF,NPIL,NFLT,I,J,K,UL,J1,MAX,SLI
     INTEGER*4 BIG
     CHARACTER*4 PC(30,2),T(150,2)
     DATA BIG/3200/

  OPEN THE DATA FILES

     OPEN(1,FILE='PILOT.DAT',STATUS='OLD')
     OPEN(2,FILE='ACCOMP.DAT',STATUS='OLD')
     OPEN(3,FILE='AVAIL.DAT',STATUS='OLD')
     OPEN(4,FILE='REQMNT.DAT',STATUS='OLD')
     OPEN(5,FILE='SCHED.DAT',STATUS='OLD')
     OPEN(6,FILE='COST.DAT',STATUS='NEW')
     OPEN(7,FILE='FEAS.DAT',STATUS='NEW')

  READ INTO THE PROGRAM THE RAW DATA FILES

     READ(1,1000) NPIL
1000 FORMAT(//I5)
     DO 5 I=1,NPIL
     READ(1,1010) (P(I,J),J=1,2),(PC(I,J),J=1,2)
1010 FORMAT(10X,2I5,3X,A2,4X,A1)
   5 CONTINUE

     READ(2,1020) (ACC(1,J),J=1,9)
1020 FORMAT(//10X,9I5)
     DO 6 I=2,NPIL
     READ(2,1025) (ACC(I,J),J=1,9)
1025 FORMAT(10X,9I5)
   6 CONTINUE

     READ(3,1030) NE(1)
1030 FORMAT(//10X,I5)
     IF(NE(1).EQ.0) GOTO 8
     DO 7 J=1,NE(1)
     READ(3,1035) (AVL(1,J,K),K=1,4)
1035 FORMAT(15X,I3,I7,I3,I7)
   7 CONTINUE
   8 CONTINUE
```

```
      DO 10 I=2,NPIL
      READ(3,'(10X,I5)') NE(I)
      IF(NE(I).EQ.0) GOTO 10
      DO 9 J=1,NE(I)
      READ(3,1035) (AVL(I,J,K),K=1,4)
    9 CONTINUE
   10 CONTINUE

      READ(4,1050) (REQ(1,J),J=1,9)
 1050 FORMAT(//10X,9I5)
      DO 20 I=2,3
      READ(4,1055) (REQ(I,J),J=1,9)
 1055 FORMAT(10X,9I5)
   20 CONTINUE

      READ(5,1060) NFLT
 1060 FORMAT(//I5)

      READ(5,1065) (ENDDAY(I),I=1,5)
 1065 FORMAT(5I5)

      DO 50 I=1,NFLT
      READ(5,1070) (T(I,J),J=1,2),(S(I,J),J=1,4)
 1070 FORMAT(6X,A4,3X,A2,I5,I3,I2,I5)
   50 CONTINUE

   END OF READING PORTION OF THE PROGRAM

   MAIN BODY OF THE PROGRAM

      WRITE(6,1100) NPIL,NFLT
 1100 FORMAT(1X,2I5)

      SLI=0
      DO 65 I=1,NPIL
      SLI=SLI+P(I,2)
      UL=P(I,1)-P(I,2)
      WRITE(6,1110) P(I,1),UL
 1110 FORMAT(1X,2I5)
   65 CONTINUE
      WRITE(6,1110) NFLT-SLI,NFLT-SLI


      CALL ARCMAT(NFLT,NPIL,ACC,AVL,REQ,PC,T,
     *NE,SCH,S,P,C)

      DO 70 J=1,NFLT+NPIL
      WRITE(6,1115) (C(I,J),I=1,NPIL+1)
 1115 FORMAT(1X,8I5)
   70 CONTINUE
```

```
      DEVELOP THE FEASIBILITY MATRIX

      NF=0
      DO 130 J=1,NFLT
      FPOINT(J)=NF+1
      MAX=J+30
      IF(MAX.GT.NFLT) MAX=NFLT
      DO 90 K=J,MAX
      IF(SCH(J,3).GE.SCH(K,1)) THEN
      NF=NF+1
      FEAS(NF)=K
      ELSE
      K=MAX
      ENDIF
   90 CONTINUE
      CREW DUTY DAYS
      J1=ENDDAY(S(J,4))
      DO 100 K=J1-12,J1
      IF ((SCH(J,1)+1200).LT.SCH(K,2)) THEN
      NF=NF+1
      FEAS(NF)=K
      ENDIF
  100 CONTINUE
      CREW NIGHTS
      IF(S(J,4).EQ.4) GOTO 130
      DO 110 K=J1+1,J1+13
      IF((SCH(J,3)+1200).GT.SCH(K,1)) THEN
      NF=NF+1
      FEAS(NF)=K
      ELSE
      K=J1+13
      ENDIF
  110 CONTINUE
  130 CONTINUE


      WRITE(7,'(1X,I5)') NF

      DO 135 I=1,35,5
      WRITE(7,1120) (FPOINT(I+J),J=0,4)
 1120 FORMAT(1X,5I5)
  135 CONTINUE


      DO 138 I=1,NF,5
      WRITE(7,1130) (FEAS(I+J),J=0,4)
 1130 FORMAT(1X,5I5)
  138 CONTINUE

      STOP
      END
 THIS SUBROUTINE DEVELOPS THE ARC MATRIX

    SUBROUTINE ARCMAT(NFLT,NPIL,ACC,AVL,REQ,PC,
   *T,NE,SCH,S,P,C)
    INTEGER*2 NFLT,NPIL,ACC(30,1),C(30,1)
    INTEGER*2 AVL(30,10,1),REQ(3,1),NE(1)
    INTEGER*2 ED,U,S(150,1),SCH(150,1),P(30,1)
    CHARACTER*4 PC(30,1),T(150,1)
    INTEGER*2 DAY1,DAY2,I,J,K1,T1,T2,BTIME,ETIME
    INTEGER*4 BIG
    DATA BIG/3200/

    DO 150 I=1,NPIL+1
    DO 140 J=1,NFLT+NPIL
    C(I,J)=3200
140 CONTINUE
150 CONTINUE

    DO 250 J=1,NFLT
    DAY1=(S(J,4)-1)*2400
    SCH(J,1)=((S(J,2)-2)*100)+DAY1+S(J,3)
    SCH(J,3)=((S(J,2)+3)*100)+DAY1+S(J,3)
    SCH(J,2)=((S(J,2)+1)*100)+DAY1
    ED=S(J,3)+30
    IF(ED .GE. 60) THEN
    ED=ED-60
    SCH(J,2)=SCH(J,2)+100
    ENDIF
    SCH(J,2)=SCH(J,2)+ED

    IF(T(J,1).EQ.'ACTT') THEN
    T1=2
    ELSEIF(T(J,1).EQ.'DACT') THEN
    T1=3
    ELSEIF(T(J,1).EQ.'DART') THEN
    T1=4
    ELSEIF(T(J,1).EQ.'NINT') THEN
    T1=5
    ELSEIF(T(J,1).EQ.'DINT') THEN
    T1=6
    ELSEIF(T(J,1).EQ.'INST') THEN
    T1=7
    ELSEIF(T(J,1).EQ.'AARD') THEN
    T1=8
    ELSE
    T1=9
    ENDIF
    DO 230 I=1,NPIL
    U=1
    IF ((PC(I,1).EQ.'WG') .AND. (T(J,2) .EQ. 'FL')) GOTO 230
    DO 200 K1=1,NE(I)
    DAY2=(AVL(I,K1,1)-1)*2400
    BTIME=DAY2+AVL(I,K1,2)
    ETIME=((AVL(I,K1,3)-1)*2400)+AVL(I,K1,4)
    IF ((ETIME.GT.SCH(J,1)).AND.(BTIME.LT.SCH(J,3))) THEN
    U=0
    K1=NE(I)
    ENDIF
200 CONTINUE

    IF (U .EQ. 1) THEN
    IF (PC(I,2).EQ.'N') THEN
    T2=1
    ELSEIF (PC(I,2).EQ.'E') THEN
    T2=2
    ELSE
    T2=3
    ENDIF
    IF((REQ(T2,T1).EQ.0).AND.(T2.NE.3)) THEN
    C(I,J)=BIG
    ELSEIF((REQ(T2,T1).EQ.0).AND.(T2.EQ.3)) THEN
    C(I,J)=(3*(ACC(I,1)*100)/REQ(T2,1))+5
    ELSE
    C(I,J)=((ACC(I,T1)*100)/REQ(T2,T1))+5
    ENDIF
    IF((PC(I,1).EQ.'FL').AND.(T(J,2).EQ.'WG'))
   *     C(I,J)=C(I,J)*2
    ENDIF
230 CONTINUE
250 CONTINUE

    DO 260 I=1,NPIL
       C(NPIL+1,NFLT+I)=((ACC(I,1)*100)/REQ(T2,1))+5
       C(I,NFLT+I)=0
260 CONTINUE
    RETURN
    END
```

```
     DAY=0
     N=1
  40 CONTINUE
     DAY=DAY+1
     WRITE(4,1100) DAY
1100 FORMAT('1','DAY ',I2)
     PER=0
     FLT=0
  50 CONTINUE
     PER=PER+1
     WRITE(4,1110) PER
1110 FORMAT('0','  PERIOD ',I2)
  60 CONTINUE
     FLT=FLT+1
     WRITE(4,1120) FLT
1120 FORMAT('0','FLIGHT',I2)
     WRITE(4,1130) TYPE(N)
1130 FORMAT('+',2X,A5)
     WRITE(4,1140) FLTN(N,2)
1140 FORMAT('+',2X,I5)
  70 CONTINUE
     WRITE(4,1150) PNAME(X(N))
1150 FORMAT('+',2X,A10)
     IF(N.EQ.NFLT) GOTO 80
     N=N+1
     IF(FLTN(N-1,3).EQ.FLTN(N,3)) GOTO 50
     IF(FLTN(N-1,4).EQ.FLTN(N,4)) GOTO 60
     IF(FLTN(N-1,5).EQ.FLTN(N,5)) GOTO 70
     GOTO 40
  80 CONTINUE
     STOP
     END
```

C.2 Optimization Program

```
      PROGRAM SOLUTN

      INTEGER*4 LB,LBSTAR,BIG,NEG,C(10,35),C1(10,35)
      INTEGER*4 R1(10),K1(35),R2(35),M(3)
      INTEGER*2 NPIL,NFLT,ITER,TAG,I,J,K,KK,K2,K3,K4
      INTEGER*2   D0,D1,D2,S0,S2,P,P1,M0,M1,Q,X0,R
      INTEGER*2 S(10,2),A(10,35)
      INTEGER*2 D(35,2),R3(35),LVAR(2)
      INTEGER*2 XSTAR(45,2),XANDY(45,2),S1(45,2),Y(45,2)
      INTEGER*4 PJ(7),SUMPJ,PJMAX,COST,MCOST
      INTEGER*2 FFEAS,BRANCH,FLAG,FFLAG,NEWMAX
      INTEGER*2 NV,L,LV,PVAR(2),SIP(7,7)
      INTEGER*2 XONE(7),FEAS(7,7),NONE,MSOL(7)
      INTEGER*2 TSOL(7),FM(7),BFEAS(7),FPOINT(35)
      INTEGER*2  F(80),LYR,CPROB(3,300),NF,START,END,QQ

      DATA BIG/3200/
      DATA NEG/-10000/
      DATA LBSTAR/100000/
      DATA LYR/0/
      DATA FFEAS/1/


   OPEN THE FILES

      OPEN(1,FILE='COST.DAT',STATUS='OLD')
      OPEN(2,FILE='OUTPUT.DAT',STATUS='NEW')
      OPEN(3,FILE='FEAS.DAT',STATUS='OLD')
      OPEN(4,FILE='BIP.DAT',STATUS='NEW')

   READ IN THE PROBLEM DATA

 9000 FORMAT(1X,A,I5)
      READ(1,1000) NPIL,NFLT
 1000 FORMAT(1X,2I5)
      S2=NPIL+1
      D1=NFLT+NPIL
      S0=0
      D0=0
      DO 10 I=1,NPIL
      READ(1,1010) S(I,1),D(NFLT+I,1)
      D(NFLT+I,2)=D(NFLT+I,1)
      S(I,2)=S(I,1)
 1010 FORMAT(1X,2I5)
   10 CONTINUE
      READ(1,1010) S(S2,1),S(S2,2)

      DO 20 J=1,D1
      READ(1,1020,END=20) (C(I,J),I=1,S2)
 1020 FORMAT(1X,8I5)
      DO 18 K=1,S2
      C1(K,J)=C(K,J)
   18 CONTINUE
   20 CONTINUE

      DO 22 I=1,NFLT
         D(I,1)=1
         D(I,2)=1
   22 CONTINUE

      READ(3,'(1X,I5)') NF
      DO 25 I=1,35,5
         READ(3,'(1X,5I5)') (FPOINT(I+J),J=0,4)
   25 CONTINUE
      DO 30 I=1,NF,5
         READ(3,'(1X,5I5)',END=30) (F(I+J),J=0,4)
   30 CONTINUE

      INITIAL SOLUTION

      D2=0
      K=1
      DO 70 I=1,NPIL
         DO 60 J=I,NFLT,NPIL
            S1(K,1)=I
            S1(K,2)=J
            D2=D2+1
            A(I,J)=D(J,2)
            S(I,2)=S(I,2)-D(J,2)
            D(J,2)=0
            K=K+1
   60    CONTINUE
         S1(K,1)=I
         S1(K,2)=NFLT+I
         D2=D2+1
         A(I,NFLT+I)=S(I,2)
         D(NFLT+I,2)=D(NFLT+I,2)-S(I,2)
         S(I,2)=0
         K=K+1
   70 CONTINUE
```

```
     DO 80 J=NFLT+1,D1+DO
        S1(K,1)=S2
        S1(K,2)=J
        D2=D2+1
        A(S2,J)=D(J,2)
        S(S2,2)=S(S2,2)-D(J,2)
        D(J,2)=0
        K=K+1
  80 CONTINUE

     TAG=0


     START TRANSPORTATION ALGORITHM

  90 CONTINUE
     TAG=TAG+1

     DUAL VARIABLE CALCULATION

     ITER=0
1140 CONTINUE
     ITER=ITER+1
     WRITE(4,1150) ITER
1150 FORMAT(1X,'ITERATION',I5)
     DO 1160 I=1,D1+DO
        K1(I)=NEG
        R2(I)=10000
1160 CONTINUE
     DO 1190 I=1,S2+S0
        R1(I)=NEG
1190 CONTINUE

     R=1
     K=1
     R1(1)=0
     K1(S1(1,2))=C(S1(1,1),S1(1,2))
     GOTO 1240
     R1(S2)=0
     DO 1200 I=D2,D1+2-S2,-1
        IF(S1(I,1).EQ.S2) THEN
           K1(S1(I,2))=C(S1(I,1),S1(I,2))
           K=K+1
           DO 1195 K=1,D2
              IF(S1(K,2).EQ.S1(I,2)) THEN
                 R1(S1(K,1))=C(S1(I,1),S1(I,2))-K1(S1(I,2))
                 R=R+1
              ENDIF
1195       CONTINUE
        ENDIF
1200 CONTINUE
```

```
1240 CONTINUE
     I=1
1250 CONTINUE
     I=I+1
     IF(K1(S1(I,2)).NE.NEG) GOTO 1300
     IF(R1(S1(I,1)).EQ.NEG) GOTO 1330
     K1(S1(I,2))=C(S1(I,1),S1(I,2))-R1(S1(I,1))
     K=K+1
1300 CONTINUE
     IF(R1(S1(I,1)).NE.NEG) GOTO 1330
     R1(S1(I,1))=C(S1(I,1),S1(I,2))-K1(S1(I,2))
     R=R+1
1330 CONTINUE
     IF(I.LT.D2) GOTO 1250
     IF(K.LT.D1+DO) GOTO 1240
     IF(R.LT.S2+SO) GOTO 1240

     FIND A VARIABLE TO PIVOT ON

     I=1
     M(1)=0
     DO 1500 R=1,S2+SO
        DO 1490 K=1,D1+DO
           IF(R.NE.S1(I,1)) GOTO 1450
           IF(K.NE.S1(I,2)) GOTO 1450
           IF((A(R,K).EQ.0).AND.
    *         (R2(K).GT.C(R,K)-R1(R)-K1(K))) THEN
              R3(K)=R
           ENDIF
           I=I+1
           GOTO 1490
1450       CONTINUE
           IF(R2(K).GT.C(R,K)-R1(R)-K1(K)) THEN
              R3(K)=R
           ENDIF
           IF(M(1).LT.C(R,K)-R1(R)-K1(K)) GOTO 1490
           M(1)=C(R,K)-R1(R)-K1(K)
           M(2)=R
           M(3)=K
1490    CONTINUE
1500 CONTINUE
     IF(M(1).GE.0) GOTO 2790
     WRITE(4,1502) ITER,M(2),M(3)
1502 FORMAT(1X,'ITER',I5,'PIVOT',2I5)

     FIND A CLOSED PATH FROM R TO K

     Y(1,1)=M(2)
     Y(1,2)=M(3)
     Q=1
     IF(M(2).EQ.S2+SO) GOTO 1960
     MO=Y(Q,1)
     M1=1
     ROW SEARCH

1610 CONTINUE
     I=0
1620 CONTINUE
     I=I+1
     IF(S1(I,1).GT.M0) GOTO 1670
     IF(S1(I,1).LT.M0) GOTO 1660
     IF(S1(I,2).GE.M1) GOTO 1720
1660 CONTINUE
     IF(I.LT.D2) GOTO 1620
1670 CONTINUE
     IF(Q.NE.1) GOTO 1830
     WRITE(4,8080)
8080 FORMAT(1X,'DEGENERATE MATRIX')
     STOP 'DEGEN'
     CHECK IF ALREADY USED
1720 CONTINUE
     X0=0
     DO 1780 J=1,Q
        IF(S1(I,1).NE.Y(J,1)) GOTO 1780
        IF(S1(I,2).NE.Y(J,2)) GOTO 1780
        X0=1
1780 CONTINUE
     IF(X0.EQ.0) GOTO 1890
     M1=S1(I,2)+1
     IF(M1.LT.D1+D0) GOTO 1660

1930 CONTINUE
     P=Y(Q,2)
     P1=Y(Q,1)+1
     Y(Q,1)=0
     Y(Q,2)=0
     Q=Q-1
     GOTO 2000
1890 CONTINUE
     Q=Q+1
     Y(Q,1)=S1(I,1)
     Y(Q,2)=S1(I,2)
     IF(Q.LE.2) GOTO 1960
     IF(Y(Q,2).EQ.M(3)) GOTO 2340
     COLUMN SEARCH

1960 CONTINUE
     P=Y(Q,2)
     P1=1
2000 CONTINUE
     K=0
2010 CONTINUE
     K=K+1
     IF(S1(K,1).LT.P1) GOTO 2040
2030 CONTINUE
     IF(S1(K,2).EQ.P) GOTO 2120
2040 CONTINUE
     IF(K.LT.D2) GOTO 2010
2050 CONTINUE
     MO=Y(Q,1)
     M1=Y(Q,2)+1
     Y(Q,1)=0
     Y(Q,2)=0
     Q=Q-1
     GOTO 1610

     CHECK FOR UNIQUE PATH SQUARE

2120 CONTINUE
     XO=0
     DO 2180 J=1,Q
        IF(S1(K,1).NE.Y(J,1)) GOTO 2180
        IF(S1(K,2).NE.Y(J,2)) GOTO 2180
        XO=1
2130 CONTINUE
     IF(XO.EQ.0) GOTO 2250
     P1=S1(K,1)+1
     IF(P1.LE.S2+SO) GOTO 2040
     GOTO 2050

     ADD STONE SQUARE TO PATH

2250 CONTINUE
     Q=Q+1
     Y(Q,1)=S1(K,1)
     Y(Q,2)=S1(K,2)
     IF(Q.LE.2) GOTO 2300
     IF(Y(Q,1).EQ.M(2)) GOTO 2340
2300 CONTINUE
     P1=Y(Q,1)+1
     MO=Y(Q,1)
     M1=1
     GOTO 1610
```

```
      FIND THE LEAST FLOW CHANGE

2340  CONTINUE
      XO=A(Y(2,1),Y(2,2))
      DO 2390 K=4,Q,2
         IF(XO.LE.A(Y(K,1),Y(K,2))) GOTO 2390
         XO=A(Y(K,1),Y(K,2))
2390  CONTINUE

      ADD AND SUBTRACT XO ALONG CLOSED PATH

2410  CONTINUE
      P=0
      DO 2450 K=1,Q,2
         A(Y(K,1),Y(K,2))=A(Y(K,1),Y(K,2))+XO
2450  CONTINUE
      DO 2630 K=2,Q,2
         A(Y(K,1),Y(K,2))=A(Y(K,1),Y(K,2))-XO
         IF(A(Y(K,1),Y(K,2)).GT.0) GOTO 2630
         IF(XO.EQ.0) GOTO 2500
         IF((Y(K,2).GT.NFLT).AND.(Y(K,1).LT.S2)) GOTO 2630
2500     CONTINUE

         I=0
         P=P+1
         IF(P.GT.1) GOTO 2630
2530     CONTINUE
         I=I+1
         IF(S1(I,1).NE.Y(K,1)) GOTO 2530
         IF(S1(I,2).NE.Y(K,2)) GOTO 2530
         WRITE(4,8050) Y(K,1),Y(K,2),XO
8050     FORMAT(1X,'PIVOT OUT',2I5,'  FLOW=',I5)
         DO 2590 J=I,D2
            S1(J,1)=S1(J+1,1)
            S1(J,2)=S1(J+1,2)
2590     CONTINUE
         S1(D2,1)=0
         S1(D2,2)=0
         D2=D2-1
2630  CONTINUE

      INSERT A NEW STONE SQUARE

      I=0
2660  CONTINUE
      I=I+1
      IF(Y(1,1).GT.S1(I,1)) GOTO 2660
      IF(Y(1,1).LT.S1(I,1)) GOTO 2700
      IF(Y(1,2).GT.S1(I,2)) GOTO 2660
2700  CONTINUE
```

```
      DO 2730 J=D2,1,-1
         S1(J+1,1)=S1(J,1)
         S1(J+1,2)=S1(J,2)
2730  CONTINUE
      S1(I,1)=Y(1,1)
      S1(I,2)=Y(1,2)
      D2=D2+1
      GOTO 1140
2790  CONTINUE
      IF(M(1).GE.0) THEN
         WRITE(4,8120)
8120     FORMAT(1X,'SOLUTION IS OPTIMAL')
      ENDIF


      OPTIMAL SOLUTION, FIND LB

      LB=0
      DO 2800 I=1,D2
         IF(C(S1(I,1),S1(I,2)).LE.0) THEN
            COST=C1(S1(I,1),S1(I,2))
         ELSE
            COST=C(S1(I,1),S1(I,2))
         ENDIF
         LB=LB+(COST*A(S1(I,1),S1(I,2)))
2800  CONTINUE
      DO 2805 I=1,NPIL
         LB=LB+((S(I,1)-D(NFLT+I,1))*C1(I,NFLT+I))
2805  CONTINUE
      WRITE(4,8100) TAG,ITER-1,LB
8100  FORMAT(1X,'TAG',I5,'  ITER',I5,'  LB=',I10)
      IF(LB.GT.BIG+100) GOTO 300
      IF(TAG.GT.40) GOTO 350



      THIS SEGMENT STARTS THE SIP SOLUTION PROCEDURE

      NEWMAX=0
      FFLAG=0
      I=0
 410  CONTINUE
      I=I+1
      NONE=0
      DO 415 K=1,7
         XONE(K)=0
 415  CONTINUE
```

```
    J=0
420 CONTINUE
    J=J+1
    IF(S1(J,1).EQ.I) THEN
    XANDY(J,1)=S1(J,1)
    XANDY(J,2)=S1(J,2)
    IF((A(S1(J,1),S1(J,2)).GT.0).AND.
   *   (S1(J,2).LE.NFLT)) THEN
       NONE=NONE+1
       XONE(NONE)=XANDY(J,2)
    ENDIF
    ENDIF
    IF(S1(J,1).LE.I) GOTO 420
    WRITE(4,'(1X,7I5)') (XONE(KK),KK=1,7)

    FILL THE SIP MATRIX

    WRITE(4,9000)'START SIP GEN',I
       DO 440 KK=1,7
          MSOL(KK)=0
          DO 430 K4=1,7
             SIP(KK,K4)=0
430       CONTINUE
440    CONTINUE

    FLAG=0
    KK=0
450 CONTINUE
    KK=KK+1
    START=FPOINT(XONE(KK))
    END=FPOINT(XONE(KK)+1)
    IF(KK.LT.NONE) THEN
    K4=KK
460 CONTINUE
    K4=K4+1
    DO 470 K3=START+1,END-1
       IF(F(K3).EQ.XONE(K4)) THEN
          FFLAG=1
          FLAG=1
          SIP(KK,K4)=1
       ENDIF
470 CONTINUE
    IF(K4.LT.NONE) GOTO 460
    ENDIF
    SIP(KK,KK)=1
    WRITE(4,'(1X,7I5)') (SIP(KK,J),J=1,7)
    IF(KK.LT.NONE) GOTO 450
    WRITE(4,9000)'END SIP MATRIX GEN',I
    WRITE(4,9000)'FLAG=',FLAG
    IF(FLAG.EQ.0) GOTO 725
    FIND THE PJ'S

    PJMAX=-5
    WRITE(4,9000)'START SIP SOLUTION',I
    MCOST=-5
    SUMPJ=0
    NV=NONE
    DO 500 J=1,NV
       K2=XONE(J)
       K3=R3(K2)
       PJ(J)=(C(K3,K2)-R1(K3)-K1(K2))-(C(I,K2)-R1(I)-K1(K2))
       SUMPJ=SUMPJ+PJ(J)
500 CONTINUE
    INITIALIZE FEAS
    DO 520 K=1,NV
       DO 510 J=1,NV
          FEAS(K,J)=0
510    CONTINUE
520 CONTINUE
    FILL IN FEAS
    DO 550 J=1,NV
    J=0
525 CONTINUE
    J=J+1
       DO 540 L=1,NV
          IF(SIP(L,J).EQ.1) THEN
             DO 530 K=1,NV
                IF(SIP(L,K).EQ.1) THEN
                   FEAS(J,K)=1
                ENDIF
530          CONTINUE
          ENDIF
540    CONTINUE
    IF(J.LT.NV) GOTO 525
550 CONTINUE

    START THE BRANCHING PROCESS

    J=0
555 CONTINUE
    J=J+1
       DO 560 K=1,NV
          TSOL(K)=0
560    CONTINUE
       TSOL(J)=J
       COST=PJ(J)
       LV=J
       FLAG=0
       DO 570 K=1,NV
          BFEAS(K)=FEAS(J,K)
          IF(BFEAS(K).EQ.0) THEN
             FLAG=1
          ENDIF
570    CONTINUE
       IF(FLAG.EQ.0) GOTO 600
       BRANCH=0

    FORWARD BRANCH

575    CONTINUE
       K=LV
580    CONTINUE
       K=K+1
       IF(BFEAS(K).EQ.0) THEN
          BRANCH=1
          TSOL(K)=K
          COST=COST+PJ(K)
          LV=K
          FLAG=0
          DO 590 K4=K,NV
             BFEAS(K4)=BFEAS(K4)+FEAS(K,K4)
             IF(BFEAS(K4).EQ.0) THEN
                FLAG=1
             ENDIF
590       CONTINUE
          K=NV
       ENDIF
       IF(K.LT.NV) GOTO 580
       IF((BRANCH.EQ.0).OR.(FLAG.EQ.0)) GOTO 600
       BRANCH=0
       GOTO 575

    BRANCH FATHOMED, CHECK FOR OPTIMUM

600    CONTINUE
       IF(COST.GT.MCOST) THEN
          MCOST=COST
          DO 610 K=1,NV
             IF(TSOL(K).GT.0) THEN
                MSOL(K)=XONE(K)
             ELSE
                MSOL(K)=0
             ENDIF
610       CONTINUE
       ENDIF

    BACKWARD BRANCH

       DO 620 K=LV+1,NV
          BFEAS(K)=BFEAS(K)-FEAS(LV,K)
620    CONTINUE
625    CONTINUE
       IF(TSOL(LV).NE.0) GOTO 630
       LV=LV-1
       IF(LV.LE.0) GOTO 670
       GOTO 625
```

```
630    CONTINUE
       IF(LV.EQ.J) GOTO 670
       TSOL(LV)=0
       COST=COST-PJ(LV)
       IF(LV.GE.NV) GOTO 600
       BRANCH=0
       GOTO 575
670 CONTINUE
    IF(J.LT.NV) GOTO 555

    WE HAVE THE OPTIMAL SOLUTION FOR THIS I

    WRITE(4,9000)'OPTIMUM FOR SIP',I
    WRITE(4,'(1X,7I5)') (MSOL(KK),KK=1,7)
    J=0
690 CONTINUE
    J=J+1
    IF(MSOL(J).EQ.0) THEN
    FLAG=0
    KK=0
700 CONTINUE
    KK=KK+1
    IF((S1(KK,1).EQ.I).AND.(S1(KK,2).EQ.XONE(J))) THEN
          XANDY(KK,1)=R3(S1(KK,2))
          XANDY(KK,2)=S1(KK,2)
          FLAG=1
    ENDIF
    IF(FLAG.EQ.1) GOTO 705
    IF(KK.LT.D2) GOTO 700
705 CONTINUE
    ENDIF
    IF(J.LE.NV) GOTO 690

    CALCULATE BOUND AND SEPARATION VARIABLE

    LB=LB+SUMPJ-MCOST
    DO 710 K=1,NV
       FLAG=0
       IF(MSOL(K).EQ.0) GOTO 710
       IF(K.EQ.NV) THEN
       DO 706 J=1,NV
          IF(FEAS(J,K).GT.0) THEN
             FLAG=FLAG+1
          ENDIF
706    CONTINUE
       ELSE
       DO 707 J=1,NV
          IF(FEAS(K,J).GT.0) THEN
             FLAG=FLAG+1
          ENDIF
707    CONTINUE
       ENDIF
```

```
         IF(FLAG.LE.1) GOTO 710
         J=0
 708     CONTINUE
         J=J+1
         IF((CPROB(1,J).EQ.I).AND.
     *      (CPROB(2,J).EQ.XONE(K))) GOTO 710
         IF(J.LT.LYR) GOTO 708
         IF(PJMAX.LT.PJ(K)) THEN
            PJMAX=PJ(K)
            PVAR(1)=I
            PVAR(2)=XONE(K)
            NEWMAX=1
         ENDIF
 710  CONTINUE
 725  CONTINUE
      WRITE(4,8200) I
8200  FORMAT(1X,'PILOT',I3,'FATHOMED')
      IF(I.LT.NPIL) GOTO 410
      IF(FFLAG.EQ.0) GOTO 280
      IF(NEWMAX.EQ.0) GOTO 300
      IF((LVAR(1).EQ.PVAR(1)).AND.
     *   (LVAR(2).EQ.PVAR(2))) GOTO 300
      LVAR(1)=PVAR(1)
      LVAR(1)=PVAR(2)

      TEST TO SEE IF XANDY IS FEASIBLE

      FFLAG=0
      I=0
      K=0
 210  CONTINUE
      I=I+1
      DO 215 R=1,7
         FM(R)=0
 215  CONTINUE
      DO 220 J=1,D2
         IF((XANDY(J,1).EQ.I).AND.(XANDY(J,2).LE.NFLT)) THEN
         K=K+1
         FM(K)=XANDY(J,2)
         ENDIF
 220  CONTINUE

      KK=0
 230  CONTINUE
      KK=KK+1
      START=FPOINT(FM(KK))
      END=FPOINT(FM(KK)+1)
      IF(KK.LT.K) THEN
      K4=KK
 240  CONTINUE
      K4=K4+1
      K3=START
```

```
  250 CONTINUE
      K3=K3+1
      IF(F(K3).EQ.FM(K4)) THEN
         FFLAG=1
         KK=K
         K3=END-1
         K4=K
      ENDIF
      IF(K3.LT.END-1) GOTO 250
      IF(K4.LT.K) GOTO 240
      ENDIF
      IF(KK.LT.K) GOTO 230
      IF((I.LT.NPIL).AND.(FFLAG.EQ.0)) GOTO 210
      WRITE(4,9000)'FFLAG XANDY=',FFLAG
      IF(FFLAG.EQ.1) GOTO 320

      CHECK TO SEE IF XANDY IS OPTIMAL TO BIP

      IF(LB.LT.LBSTAR) THEN
         LBSTAR=LB
         DO 270 J=1,D2
            XSTAR(J,1)=XANDY(J,1)
            XSTAR(J,2)=XANDY(J,2)
  270    CONTINUE
      ENDIF
      IF(FFEAS.EQ.1) GOTO 350
      GOTO 300

      CHECK IF S1 IS OPTIMAL

  280 CONTINUE
      IF(LB.LT.LBSTAR) THEN
         LBSTAR=LB
         DO 290 J=1,D2
            XSTAR(J,1)=S1(J,1)
            XSTAR(J,2)=S1(J,2)
  290    CONTINUE
      ENDIF
      IF(FFEAS.EQ.1) GOTO 350

      OVERALL BRANCH AND BOUND CONTROL

      ELIMINATE VARIABLES

  300 CONTINUE
      WRITE(4,8030) TAG
 8030 FORMAT(1X,'TAG',I5,'  ELIMINATE VARS')
  310 CONTINUE
```

```
      IF(LYR.EQ.0) GOTO 350
      FLAG=0
      J=CPROB(1,LYR)
      K=CPROB(2,LYR)
      IF(CPROB(3,LYR).EQ.0) THEN
         C(J,K)=C1(J,K)
         CPROB(1,LYR)=0
         CPROB(2,LYR)=0
         LYR=LYR-1
         FLAG=1
      ELSE
         C(J,K)=BIG
         CPROB(3,LYR)=0
      ENDIF
      IF(FLAG.EQ.1) GOTO 310
      GOTO 90

      ADD NEW VARIABLES

  320 CONTINUE
      WRITE(4,8040) TAG,PVAR(1),PVAR(2)
 8040 FORMAT(1X,'TAG',I5,'  ADD VAR',2I5)
      LYR=LYR+1
      CPROB(1,LYR)=PVAR(1)
      CPROB(2,LYR)=PVAR(2)
      CPROB(3,LYR)=1
      C(PVAR(1),PVAR(2))=-3200
      ADD ZERO VARIABLES
      IF(PVAR(2).LT.11) THEN
         QQ=PVAR(2)-1
      ELSE
         QQ=10
      ENDIF
      DO 327 K=PVAR(2)-QQ,PVAR(2)-1
         DO 323 J=FPOINT(K)+1,FPOINT(K+1)-1
            IF(F(J).EQ.PVAR(2)) THEN
               LYR=LYR+1
               CPROB(1,LYR)=PVAR(1)
               CPROB(2,LYR)=K
               CPROB(3,LYR)=0
               C(PVAR(1),K)=BIG
            ENDIF
  323    CONTINUE
  327 CONTINUE
```

```
      START=FPOINT(PVAR(2))
      END=FPOINT(PVAR(2)+1)
      DO 330 J=START+1,END-1
         I=PVAR(1)
            LYR=LYR+1
            CPROB(1,LYR)=I
            CPROB(2,LYR)=F(J)
            CPROB(3,LYR)=0
            C(I,F(J))=BIG
  330 CONTINUE
      GOTO 90

      OPTIMAL SOLUTION IS REACHED

  350 CONTINUE
      DO 360 I=1,D2
         IF((A(XSTAR(I,1),XSTAR(I,2)).GT.0).AND.
     *      (XSTAR(I,2).LE.NFLT)) THEN
         WRITE(2,'(1X,3I10)') XSTAR(I,1),XSTAR(I,2),
     *                        A(XSTAR(I,1),XSTAR(I,2))
         ENDIF
  360 CONTINUE
      LB=0
      DO 370 I=1,D2
         LB=LB+(C1(XSTAR(I,1),XSTAR(I,2))*A(XSTAR(I,1),
     *            XSTAR(I,2)))
  370 CONTINUE
      WRITE(2,8000) LBSTAR
 8000 FORMAT(1X,I10,' = LBSTAR')
      WRITE(2,8010) TAG
 8010 FORMAT(1X,I10,' = NO. OF ITERATIONS')
      STOP
      END
```

C.3 Program to Format Schedule

```
     PROGRAM OUTPUT

     INTEGER*2 FIL(300,2),FLTN(150,5),NUMF,X(150)
     INTEGER*2 FLAG,NPIL,NFLT,PER,DAY,FLT,N
     CHARACTER*4 TYPE(150),PNAME(35)

     OPEN(1,FILE='OUTPUT.DAT',STATUS='OLD')
     OPEN(2,FILE='PILOT.DAT',STATUS='OLD')
     OPEN(3,FILE='SCHED.DAT',STATUS='OLD')
     OPEN(4,FILE='BYNAME.DAT',STATUS='NEW')

     READ(2,1000) NPIL
1000 FORMAT(//I5)
     READ(3,'(//I5)') NFLT

     DO 10 I=1,NPIL
     READ(2,1020) PNAME(I)
1020 FORMAT(A10)
  10 CONTINUE

     DO 20 J=1,NFLT
     READ(3,1030) TYPE(J),(FLTN(J,K),K=1,5)
1030 FORMAT(3X,A5,5X,5I5)
  20 CONTINUE

     DO 30 K=1,NFLT
     READ(1,1040) FIL(K,1),FIL(K,2)
1040 FORMAT(1X,2I10)
  30 CONTINUE

     FLT=0
  35 CONTINUE
     FLT=FLT+1
     K=0
     FLAG=0
  37 CONTINUE
     K=K+1
     IF(FIL(K,2).EQ.FLT) THEN
        X(FLT)=FIL(K,1)
        FLAG=1
     ENDIF
     IF(FLAG.EQ.1) GOTO 35
     IF(K.LT.NFLT) GOTO 37
     IF(FLT.LT.NFLT) GOTO 35
```

BIBLIOGRAPHY

1. Arabeyre, J.P., Fearnley, J., Steiger, F.C., and Teather, W., "The Airline Crew Scheduling Problem: A Survey", Transportation Science, Vol. 3, pp. 140-163, (1969).

2. Arthur, J.L. and Ravindran, A, "A Multiple Objective Nurse Scheduling Model," AIIE Transactions, Vol. 13, No. 1, pp. 55-60, (1981).

3. Balinski, M., "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12, pp. 253-313, (1965).

4. __________ and Quandt, R., "On an Integer Program for a Delivery Problem," Operations Research, Vol. 12, pp. 300-304.

5. Bertsekas, D.P., "A New Algorithm for the Assignment Problem", Math Programming, North-Holland Publishing Co., Vol. 21, pp. 152-171, (1981).

6. _______________, "A Unified Framework for Primal-Dual Methods in Minimum Cost Network Flow Problems", Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems Working Paper, Massachusetts Institute of Technology, (1982).

7. Bradley, S.P., Hax, A.C., and Magnanti, T.L., Applied

Mathematical Programming, Addison-Wesley Publishing Co., Reading, MA, 1977.

8. Cohon, J.L., Multiobjective Programming and Planning, Academic Press, New York, NY, 1978.

9. Conway, R.W., Maxwell, W.L., and Miller, L.W., Theory of Scheduling, Addison-Wesley Publishing Co., Reading, MA, 1967.

10. Erlenkotter, D., "A Dual Based Procedure for Uncapacitated Facility Location", Operations Research, Vol. 26, No. 6, pp. 992-1009, (1978).

11. Etcheberry, J., "The Set-Covering Problem: A New Implicit Enumeration Algorithm", Operations Research, Vol. 25, pp. 760-772, (1977).

12. Fisher, M.L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", Management Science, Vol. 27, No. 1, pp. 1-18, (1981).

13. ____________, Jaikumar, R., and Van Wassenhove, L.N., "A Multiplier Adjustment Method for the Generalized Assignment Problem", Department of Decision Sciences Working Paper, University of Pennsylvania, (1981).

14. Ford, L.R. and Fulkerson, D.R., Flows in Networks, Princeton University Press, Princeton, NJ, 1962.

15. Garfinkel, R.S. and Nemhauser, G.L., "The Set Partitioning Problem: Set Covering with Equality Constraints," Operations Research, Vol. 17, No. 5, pp. 848-856, (1969).

16. ______________ and ______________, Integer Programming, John Wiley and Sons, New York, NY, 1972.

17. Golden, B.L. and Magnanti, T.L., Network Optimization, John Wiley and Sons (to appear).

18. Held, M., Wolfe, P., and Crowder, H.P., "Validation of Subgradient Optimization", Math Programming, North-Holland Publishing Co., Vol. 6, pp. 62-68, (1974).

19. Hogan, W.W., Marsten, R.E., and Blankenship, J.W., "The Boxstep Method for Large Scale Optimization", Operations Research, Vol. 23, p. 3, (1975).

20. Ignizio, J.P., Goal Programming and Extensions, D.C. Heath and Co., Lexington, MA, 1976.

21. Kennington, J.L. and Helgason, R.V., Algorithms for Network Programming, John Wiley and Sons, New York, NY, 1980.

22. Laffin, J., Fight for the Falklands, St. Martins Press, New York, NY, 1982.

23. Levin, R.I., Kirkpatrick, C.A. and Rubin, D.S.,

Quantitative Approaches to Management, McGraw-Hill, New York, NY, 1982.

24. Loomba, N.P. and Turban, E., Applied Programming for Management, Holt, Rinehart, and Winston, New York, NY, 1974.

25. Magnanti, T.L., "Optimization for Sparse Systems", Sparse Matrix Computations, D.Rose and J. Bunch, eds., Academic Press, New York, NY, 1976.

26. Marsten, R.E., "An Algorithm for Large Set Partitioning Problems", Management Science, Vol. 20, No. 5, pp. 774-787, (1974).

27. Miller, H.E., "Personnel Scheduling in Public Systems: A Survey", Socio-Economic Planning Sciences, Vol. 10, No. 6, pp. 241-249, (1976).

28. ________, Pierskalla, W.P. and Rath, G.J., "Nurse Scheduling Using Mathematical Programming", Operations Research, Vol. 24, No. 5, pp. 857-870, (1976).

29. Muth, J.F. and Thompson, G.L., eds., Industrial Scheduling, Prentice-Hall, New York, NY, 1963.

30. Moreland, J.A., "Scheduling of Airline Flight Crews", MS thesis, Department of Aeronautics and Astronautics, Sept 1966.

31. Nanney, T.R., Computing: A Problem Solving Approach with Fortran 77, Prentice-Hall, Englewood Cliffs, NJ, 1981.

32. Nickoletti, B., "Automatic Crew Rostering", Transportation Science, Vol. 9, No. 1, pp. 33-42, (1975).

33. Pierce, J.F., "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems", Management Science, Vol. 15, pp. 191-209, (1968).

34. Poole, L., ed., Practical Basic Programs, Osborne/ McGraw-Hill, Berkeley, CA, 1980.

35. Ross, G.T. and Soland, R.M., "A Branch and Bound Algorithm for the Generalized Assignment Problem", Math Programming, Vol. 8, pp. 91-103, (1975).

36. Rubin, J., "A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling", Transportation Science, Vol. 7, No. 1, pp. 34-48, (1973).

37. Shapiro, J.F., "Dynamic Programming Algorithms for the Integer Programming Problem - I: The Integer Programming Problem Viewed as a Knapsack Type Problem", Operations Research, Vol. 16, NO. 1, pp. 103-121, (1968).

38. ____________, Mathematical Programming, Structures and Algorithms , John Wiley and Sons, New York, NY, 1979.

39. ____________, "A Survey of Lagrangian Techniques for Discrete Optimization", Annals of Discrete Mathematics, Vol. 5, pp. 113-138, (1979).

40. Shepardson, W.B., "A Lagrangian Relaxation Algorithm for the Two-Duty Period Scheduling Problem", Ph.D Dissertation at the Massachusetts Institute of Technology, Alfred P. Sloan School of Management, June 1978.

41. <u>Tactical Air Command Manual</u> 51-50, Vol. I, Department of the Air Force, Headquarters Tactical Air Command, Langley AFB, VA, 26 Oct. 1981.

FILM